جزوه برنامه سازی پیشرفته – استاد خلیل زاده

شرکت Sun در سال ۱۹۹۹ اجازه استفاده از زبان برنامه نویسی Java را در اختیار شرکت Microsoft قرار داد، همان طور که می دانید Java به هیچ Platform یا سیستم عامل خاصی وابسته نبود. Microsoft بخشی از مفاد قرارداد نامه را زیر پا گذاشت و قابلیت مستقل از سیستم عامل بودن را از Java برداشت. شرکت Sun نیز پرونده ای علیه او تشکیل داد و Microsoft مجبور شد تا زبان برنامه نویسی جدیدی با کامپایل جدید که به C++ شبیه بود درست کند. در طول ساخت Net. کلاسهای کتابخانه ای با زبان و کامپایل SMC نوشته شدند. در سال ۱۹۹۹ *آندرس هلزبرگ* گروهی را برای طراحی زبانی جدید که در آن زمان نامش Cool[1] بود را تشکیل دادند. ولی به دلیل مناسب نبودن اهداف تجاریِ این نام، در معرفی رسمی آن در سال ۲۰۰۰ نامش از Cool به C# تغییر یافت. مدیر و سرپرست طراحان در Microsoft آندرس هلزبرگ بوده که تجره ی قبلی او در طراحی Framework و زبان های برنامه نویسی Delphi، Borland++ و Visual C بوده است.

C# یک زبان برنامه سازی ساده، مدرن، برای اهداف عمومی و شیء گرا می باشد. به دلیل اهمیت داشتن موضوع نیرومندی و بهره وری برنامه نویسی زبان دارای چک کننده، جک کننده مرز های آرایه، تشخیص حالت هایی که یک متغیر مقدار دهی اولیه نشده است و قابلیت انتقال کد ها میباشد. این زبان برای استفاده در اجزای توسعه ی نرم افزار برای دستیابی به مزایای سیستم های توزیع شده در نظر گرفته شده است. قابلیت انتقال برنامه نویس بسیار مهم مهم میباشد به خصوص برای کسانی که به زبان های برنامه نویسی C و C++ آشنایی دارند. برنامه های نوشته شده با C# طوری هستند که از لحاظ حافظه و پردازنده ی مورد نیاز مقرون به صرفه باشند. (قابلیت انتقال برنامه نویسی یعنی if و for و ... کاملا شبیه C است)

زبان C# یک زبان شیء گرا و سطح بالا از خانواده ی زبان های چارچوب Net. در شرکت Microsoft می باشد. زبان C# یک زبان برنامه نویسی چند الگویی است و منظم شده ی مدل های تابعی، شیء گرا و جزء گرا میباشد. C# یکی از چهار زبانی است که توسط CLR از Net Framework. پشتیبانی میشود و در همه جا با نام Microsoft Visual Studio شناخته میشود. برخی از تفاوت های C# با C و C++ عبارت اند از اینکه هیچ تابع یا متغیر سراسری وجود ندارد. تمام متد ها و اعضا باید در کلاس تعریف شوند. هر چند برای استفاده از متغیر ها و توابع عمومی باید از متد ها و متغیر ها در کلاس های عمومی استفاده شود. C# دارای یک داده ی از جنس Boolian هست. برخی عبارت ها مانند if و While که شرطی هستند نیازمند یک عبارت از جنس Boolian هستند. همانطور که C++ هم دارای نوع داده ی Boolian میباشد. این نوع داده به راحتی می تواند به و یا از Integer ها تبدیل شود. و عبارتی مانند if(a) نیازمند این امرست که a از یک نوع قابل تبدیل به Boolian یا اشاره‌گر باشد. کامپایلر C# در این شرایط برنامه نویس را مجبور به استفاده از عبارتی می کند که به درستی یک مقدار Boolian را بر میگرداند. بنابراین دستوری مانند if(i=j) باعث بروز خطا میشود و درست آن if(i==j) است.

[1] C Like Object Oriented Language

وراثت چندگانه از کلاسها در این زبان پشتیبانی نمیشود. البته کلاسها امکان ارثبری از تعداد نامحدود واسطها را دارا میباشند. پشتیبانی نکردن از وراثت چندگانهبه دلیل اهداف معماری در این زبان در CLI و برای جلوگیری از پیچیدگی بیشتر میباشد.

تبدیل b به a در حالتی که a یک integer و b یک float باشد در زبان C++ مجاز است. اما در C# به یک خطای زمان کامپیال منجر میشود.

شیءگرایی در C# بر چند پایه استوار است:

۱- **ارثبری (Inheritance):** پدر و فرزندی را درنظر بگیرید. هر پدری مشخصات فردی به خصوصی دارد. فرزند میتواند تمامی خصوصیات پدر را به ارث ببرد. علاوه بر آن دارای یک سری خصوصیات دیگری نیز باشد. برای مثال وقتی پدر عصبانی میشود فریاد میکشد. فرزند نیز در هنگام عصبانیت فریاد میکشد اما علاوه بر آن چند عدد بشقاب نیر میشکند. به عنوان مثالی دیگر دوچرخه را در کلاس وسیله های نقلیه فرض کنید. همه ی خصوصیات مربوط به وسایل نقلیه را دارا میباشد و علاوه بر آن خصوصیات مختص خود را نیز دارد. قابلیت استفاده ی دوباره از کد یکی از خصوصیات ارثبری میباشد

۲- **کپسوله سازی (Encapsulation):** همانطور که از اسمش پیداست به قرارداد پیاده سازی در یک کپسول اشاره میکند. به طوری که کاربر بیرونی از نحوه ی پیاده سازی مطلع نباشد و فقط بداند این کپسول کار خاصی را انجام میدهد. وقتی که یک کپسول می خورید نمی دانید داخل آن چه چیزی قرار دارد و فقط به این فکر میکنید که این کپسول چه تاثیری در بدن شما دارد. هدف کپسوله سازی این است که ما را از پرداختن به زیر موضوعات رها کند و اشیاء را به صورت یک جعبه ی سیاهی بدانیم که به ازای یک ورودی خاص خروجی خاصی به ما میدهند.

۳- **چند ریختی (Polymorphism):** فرض کنید پدر کار خاصی را به طریق خاصی انجام می دهد. مثلاً برای پختن غذا ابتدا ظرف ها را شسته سپس گاز را روشن می کند و سپس غذا را میپزد. فرزند که خصوصیات و کارهای او را به ارث برده است برای پختن غذا ابتدا گاز را روشن میکند سپس کبریت می زند و بعد غذا را میپزد و بعد ظرف ها را می شوید. برادر نیز ممکن است همین کار ها را به روش دیگری انجام دهد. یعنی یک کار ثابت به طرق مختلفی انجام میشود. به این قضیه، چند شکلی میگویند

۴- **مجرد سازی (Abstraction):** به کلاسی مجرد گفته میشود که پیاده سازی متدها در آن انجام نمیشود. فرض کنید شما رئیس یک شرکت بزرگ برنامه نویسی هستید و می خواهید پروژه ی بزرگی را انجام دهید. برای اجرای پروژه ار برنامه نویسان مختلفی استفاده میکنید که ممکن است همه آنها هموطن نباشند. مثلاً روسی، ایرانی، هندی، آلمانی، انگلیسی و فرانوسی باشند. اگر قرار باشد هر برنامه نویسی در نام گذاری متد ها و کلاسهایش آزاد باشد در کدنویسی هرج و مرج به

وجود می‌آید. شما نیز به عنوان مدیر پروژه کلاسی را تعریف می کنید که در آن تمام متد ها با ورودی و خروجی هایتان مشخص می شود.

۵- **Interface**: در برنامه نویسی همانند کلاس است. تنها با این تفاوت که هیچ کدام از اعضای آن پیاده سازی نمی شوند. در واقع یک Interface گروهی از متدها، خصوصیات، رویدادها هستند که در کنار هم جمع شده اند، Interface ها را نمیتوان وهله سازی کرد. تنها چیزی که یک Interface دارد، امضای تمامی اعضای آن میباشد. به این معنی که تمامی ورودی ها و خروجی های آن در آن تعریف میشوند ولی چیزی پیاده سازی نمیشود. Interface ها بسیار شبیه کلاسها هستند تنها با این تفاوت که در Interface ها پیاده سازی وجود ندارد. Java و C# از ارثبری چند گانه پشتیبانی نمی کنند. (بر خلاف C++) یعنی یک کلاس از چند کلاس دیگر به ارث نمی برد. گاهی لازم داریم از چند کلاس به ارث ببریم. راه حلش این است که از Interface ها استفاده کنیم. ولی بدانید که اگر از Interfaceی به ارث ببریم باید تمام متد های آن را به کار ببریم. یک کلاس میتواند از n تا Interface و تنها یک کلاس به ارث ببرد.

نحوه ی ساخت جدول در Access:

در قسمت File گزینه ی New را انتخاب کرده و در سمت راست نرم افزار در قسمت Blank Database نام پایگاه داده ی خود را مشخص می کنیم. مثلا DBA و سپس دکمه ی Create را انتخاب می کنیم. در صفحه ی جدید بر روی گزینه ی Table1:Table کلیک کرده و گزینه ی Design View را انتخاب می کنیم. نام جدول را انتخاب می کنیم. در صفحه ی جدید سه ستون به نام های Field Name، Data Type و Description وجود دارد. در قسمت Fieldname فیلد های مورد نظر را وارد میکنیم. به عنوان مثال فیلد هایی که برای دانشجویان یک دانشگاه مد نظر است شامل نام، نام خانوادگی، شماره دانشجویی، رشته، معدل، سال ورود، سن و آدرس میباشد. همان طور که می بینید در قسمت Fieldname اولین فیلد به نام ID به صورت خودکار ایجاد شده است که نوع آن Auto Number است. فیلد های مورد استفاده ما به قرار زیر میباشد:

First name (text)	Lastname (text)	StudentNo (text)
Category (text)	Average (text)	Arrival Year (text)
Age (text)	Address (text)	

در ادامه بر روی دکمه ی View رفته و گزینه ی Datasheet View را بزنید. در قسمت File، Save as را انتخاب کنید و فایل را در فرمتِ Access 2000 database ذخیره کنید.

نحوه ساخت جدول در SQL:

در منوی سمت چپ گزینه ی اول که Database میباشد را انتخاب کرده، بر روی آن کلیک راست می کنیم و New Database را انتخاب می کنیم. در اینجا نام Database مورد نظر را می نویسیم و بعد

Ok می‌کنیم. نام را DBA در نظر بگیرید. در قسمت Tables برای دیتابیسِ DBA کلیک راست کـرده و گزینه ی Table New را انتخاب می کنیم. سه ستونِ زیر نمایان میشوند:

Column Name	Data Type	Allow Nulls

همان طور که میدانید در Access فیلد ID از نوع Auto Number به صورت خودکار ساخته میشود. امـا در SQL به صورت خودکار این خاصیت وجود ندارد و روش ساختن فیلد ID از نوع Auto Number بـه این صورت است که ابتدا فیلدِ ID را از نوع INT تعیین می کنیم و در پنجـره پـایین صـفحه (Column Properties) در قسمتِ Identity مقدارِ Identity is را برابر Yes قرار می‌دهیم. فیلدهای بیان شده در بخش قبل را به این نرم افزار هم اضافه میکنیم و بخش Data Type مربوط بـه همـه ی فیلـد هـا را برابـر nvarchar(50) قرار می‌دهیم. نهایتاً Save کرده و نام جدول را نیز DBA قرار می دهیم.

آشنایی با محیط Visual Studio:

وقتی برای اولین بار نرم افزار را اجرا می کنید در ابتدای کار نوع زبان برنامه نویسی که می خواهیم اسـتفاده کنیم را از ما سوال می کند. (C# یا VB یا J# یا VC++)

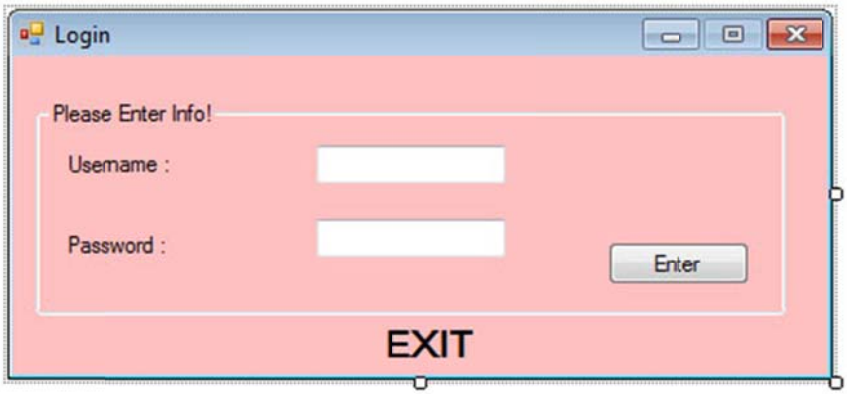
در منوی Tools بـر روی گزینـه ی Import & Export Settings کلیـک مـی‌کنـیم. در ایـن قسـمت میتوانیم نحوه ی نمایش پنجره ها و تنظیمات محیط VS را توسط گزینه اول ذخیره کنیم. توسط گزینه ی دوم نیز آنرا باز می‌گردانیم تا تنظیمات و نحوه ی نمایش به حالتی کـه ذخیـره کـرده ایـم برگـردیم. توسـط گزینه سوم تنظیمات به حالت اولیه کارخانه ای باز می گردد.

جهت ایجاد یک پروژه ی جدید از ۳ روش زیر می توان استفاده کرد:

۱- به گزینه ی File رفته و در قسمت New گزینه ی Project را انتخاب میکنیم.
۲- با استفاده از آیکون New Project
۳- کلید های Ctrl + Shift + N

در سمت چپ (Project Type) گزینه هایی ماننـد Windows، Web، Smart Device و ... مشـاهده می شود. Windows برای این بکار می رود که از برنامه هایی تحـت Windows Form Application استفاده نماییم و Web برای نوشتن برنامه هایی تحت Asp.Net و Smart Device برای نوشتن برنامـه هـایی تحـت Mobile اسـتفاده میشـود. بـا انتخـاب Windows و گزینـه ی Windows Form Application وارد محیط برنامه نویسیمان میشویم. لازم است تا نام و مکان آنرا تعیین کنیم.

یک فرم ساخته میشود. در پنجره ی سمت راست، نام آن را Login و Form Border Style آنرا بر روی Fixed Single قرار میدهیم. برای اجرای برنامه میتوانیم از دکمه ی F5 استفاده کنیم. در ادامه چند شیء قرار میدهیم. یک عدد Button که Text آن مقدار Enter است، دو عدد TextBox که مقدار Text Align آنها Center میباشد. برای TextBox دوم که مربوط به رمز میباشد، مقدار Use System Password Char را برابر Yes قرار می دهیم. سه عدد Label به نام های Username، Password و Exit می سازیم و کل این اشیاء را داخل یک Group Box قرار می دهیم و در قسمت Text آن عبارت Please Enter info! را وارد می‌کنیم. میتوانیم با ۲ بار کلیک کردن بر روی هر شیء وارد محیط برنامه نویسی اش بشویم.



جهت ایجاد فرم جدید از قسمت Solution Explorer بر روی نام برنامه کلیک راست کرده و در قسمت Add بر روی Windows Form کلیک می کنیم. در پنجره ی ایجاد شده در قسمت Name مقدار Form2.cs را - که cs مربوط به کلمه کلاس میباشد - به مقدار View تغییر می دهیم. به همین ترتیب فرم های دیگری به نام های Search، Add، Delete و Update ایجاد می کنیم.

نحوه ی تعریف یک کلاس: برای تعریف یک کلاس از متد زیر استفاده می کنیم:

نام کلاس	نام نمونه = new	نام کلاس ();	
برای مثال:	f = new	Select ();	Select

کلمه ی New: هنگامی که شما یک متغیر از نوع کلاس تعریف می کنید این متغیر یک شیء نیست بلکه در واقع یک ارجاع هست که می تواند به یک شیء اشاره کند و شیء واقعی با استفاده از کلمه کلیدی New ایجاد میشود. کاربرد کلمه کلیدی New اینست که این کلمه در زمان اجرا حافظه مورد نیاز برای ایجاد شیء را از سیستم می گیرد و یک ارجاع به آن باز می گرداند که این ارجاع می تواند در متغیر تعریف شده ذخیره شود.

به فرم اولیه ساخته شده بر می گردیم و روی دکمه ی Enter که ساخته ایم دو بار کلیک کرده و برنامه زیر را می نویسیم:

```
this.Hide();
string a, b;
a = "uni";
b = "rud";
View frm = new View();
if (textBox1.Text == a && textBox2.Text == b)
   frm.ShowDialog();
if (textBox1.Text != a || textBox2.Text != b)
   MessageBox.Show("Please Fill Correct Info!!", "Error");
```

بر روی نوشته ی Exit که در فرم ایجاد کرده ایم ۲ کلیک کرده و برنامه زیر را می نویسیم:

```
Application.Exit();
```

باید کردن جدول های Access و SQL:

در روش بایند کردن تمامی داده های جدول پایگاه داده مان را می توان به کنترل خاصی (شیءِ خاصی) در محیط برنامه‌نویسی C# واکشی کرد. این کار را میتوان بدون کدنویسی در مرحله اول انجام داد:

برای مثال میتوان شیءِ Data Grid View را در یک فرم ایجاد کرده و در آن جدول مورد نظر را بایند کرد. به این صورت که در سمت راست Data Grid View در قسمت بالای آن مثلث را که قرار دارد انتخاب می کنیم تا پنجره ی Data Grid View Task باز شود. در این پنجره در قسمت Choose Data Source (انتخاب منبع داده) دیتابیسِ مورد نظر را با کلیک بر Add Project data source انتخاب کرده و در پنجره ی ایجاد شده با انتخاب گزینه ی Database به مرحله بعد رفته و با کلیک بر روی دکمه ی New Connection پنجره ی انتخاب منبع داده برای ما نمایش داده میشود که با توجه به جدول مورد نظر (Access، SQL یا Oracle) انتخابمان را انجام میدهیم:

- گزینه ی Microsoft Access Database File را انتخاب می کنیم.
- در قسمت Database Filename با انتخاب دکمه ی Browse جدول مورد نظر را نیز انتخاب می‌کنیم.
- برای Edit جدول واکشی شده در Data Grid View با انتخاب گزینه ی Edit Columns تغییرات مورد نظر را در قسمت Selected Columns بر روی هر ستون می توانیم انجام دهیم.

بایند کردن SQL هم به همین گونه است. با این تفاوت که در مرحله انتخاب نـوع پایگـاه داده بایـد SQL Database Server را انتخاب کنید. تنها تفاوت SQL و Access در بایند کردن این است که در قسمت Server name با انتخاب کـاراکتر نقطـه میتـوان بـه قسـمت Select or Enter a Database Name دسترسی یابیم.

ساخت جدول در اوراکل:

پس از نصب نرم افزار Oracle و Maestro با انتخاب نـام دیتـابیس در نـرم افـزار Maestro مـی تـوانیم جدولی ایجاد کنیم. با انتخاب گزینه Create New Table در فرم باز شده جدید نیز نام جدول را انتخاب کرده و در فرم بعدی با رایت کلیک بر صفحه و انتخاب گزینه ی Add New Field فیلـد هـای جدیـد را ایجاد می کنیم.

کلاس ها و اشیاء:

کلاس الگویی برای اشیاء با ویژگی ها و صفات یکسان می باشند. بعنـوان مثـال کـلاس حیوانـات را در نظـر بگیرید که ویژگی تمام حیوانات را داشته باشد مثلا غذا خوردن، بیدار شدن، خوابیدن، مردن، حرکت کـردن و ... این کلاس بعنوان یک الگو برای حیوانات است.کلاس حیوانات برای تعریف حیوانات مختلف بکار می رود که هر حیوان می تواند ویژگی های خاص خود را داشته باشد: ۱- خزنده است یـا پرنـده ۲- اهلـی اسـت یـا وحشی و ... برای استفاده از کلاس باید ۲ کار انجام دهیم: ۱- تعریف کلاس ۲- نمونه سازی از کلاس

برای تعریف کلاس به صورت زیر اقدام میکنیم:

```
سطح دسترسی کلاس    Class    نام کلاس
{
اعضای کلاس
}
```

سطح دسترسی کلاس تعیین می کند که کلاس تعیین شده در کجا می تواند استفاده شود. سـطح دسـترسی می تواند Public یا Internal باشد. Public کلاس را طوری تعریف می کند که در خارج از فضـای نـامی که در آن تعریف شده است قابل استفاده باشد و Internal کلاس را طوری تعریـف مـی کنـد کـه فقـط در فضای نام تعریف شده قابل دسترسی باشد.

نمونه سازی کلاس: همانطور که قبلاً گفته شد برای استفاده از کلاس باید اشیائی از نوع آن ایجاد کرد:

```
نام کلاس    نام نمونه = new    نام کلاس () ;
```

اعضای کلاس: عبارت اند از: **۱**- ثابت ها **۲**- فیلد ها **۳**- خواص **۴**- متد ها **۵**- سازنده ها **۶**- مخرب ها و ...

ثابت ها: مقادیری هستند که قابل تغییر نمی باشند.

سطح دسترسی const نوع متغیر نام ثابت = مقدار ;

برای مثال:

```
Private cons tint x=5;
```

فیلدها: همانند متغیر معمولی برای ذخیره کردن داده ها بکار میروند. برای مثال:

```
Public int x;
Private float y;
```

دستور اول فیلدی به نام x با دستیابی عمومی (در خارج از کلاس میتوان آن را دستیابی کرد) دستور دوم فیلدی به نام y از نوع اعشاری (Float) با دستیابی خصوصی را تعریف می کند.

خواص: متدهایی هستند که داده های فیلد ها را مقدار دهی کرده یا مقدار آنها را بازیابی می کنند. برای مثال مقدار دهی از واژه ی Set و برای بازیابی مقدار خاصیت از واژه ی Get استفاده می شود.

مثال:

```
Public Float y;
{
Get {return y;}
Set {y=Value;}
}
```

متدها: مجموعه دستوراتی هستند که عمل خاصی را به روی کلاس انجام می دهند. برای استفاده از متد باید تعرف متد و فراخوانی آن را به روش زیر انجام داد.

```
سطح دسترسی  نوع مقدار برگشتی متد  نام متد  (لیست پارامتر های مجاز)
{
دستورات بدنه;
}
```

سازنده ها: وقتی نمونه ای از کلاسها ایجاد می کنیم، سازنده کلاس ایجاد می شود. سازنده کلاس فراخوانی می‌شود. با استفاده از سازنده می توانیم در هنگام ایجاد نمونه، اعضای داده یک کلاس را مقدار دهی اولیه کنیم. یک کلاس می تواند هیچ سازنده، یک سازنده یا چند سازنده داشته باشد.

دستیابی به بانک اطلاعاتی با ADO.Net: همان طور که بیان گردید، همان فایلهای کامپیوتر هستند که برنامه های کاربردی با استفاده از سیستم مدیریت بانک اطلاعات، آن را پردازش می کند. اما برای اینکه

برنامه کاربردی با سیستم مدیریت بانک اطلاعاتی (DBMS) ارتباط بر قرار کند، نیاز به یک واسط نرم افزاری دارد. یکی از این واسطهای نرم افزاری ADO.Net می باشد. به عبارت دیگر زبانهای C# و VB برای برقراری ارتباط با سیستم مدیریت بانک اطلاعاتی، از فن آوری ADO.Net استفاده می کند.

نکته: توجه داشته باشید که فناوری ADO با فناوری ADO.Net کاملاً متفاوت است زیرا الگوی بسیار جدیدی برای دستیابی به بانک اطلاعات دارد. ADO.Net امکان برقراری ارتباط با بانک اطلاعات رابطه ای و سایر منابع داده را فراهم می کند. به عبارت دیگر ADO.Net یک فناوریست که برنامه کاربردی Net. از آن برای ارتباط به بانک اطلاعات استفاده می کند.

تذکر: برای برنامه های کاربردی توزیعی مفید است. (مثل کاربردی Web)

به این ترتیب می توانید Record هایی را به بانک اطلاعات حذف و اضافه کنید یا تغییر دهید. یکی از ویژگی های ADO.Net این است که Connectionless است. این ویژگی تفاوت اساسی آن با ADO است. در ADO برنامه ی کاربردی به بانک اطلاعات وصل می شود، یک Record Set ایجاد می کند، و از اطلاعات آن برای پر کردن Data Grid View یا محاسبات دیگر استفاده می کند. سپس Record Set را حذف کرده و اتصال ر ا قطع می کند. در حالی که اتصال باز است (اتصال با Database برقرار است) اتصال فعال با Database وجود دارد که می توانیم آن را فوراً به‌هنگام‌سازی (Update) کنیم و گاهی تغییرات حاصل از کاربران دیگر را مشاهده کنیم. در یک برنامه نه چندان خوب اتصال بانک اطلاعات ممکن است در حین انجام کارهای دیگر، باز نگه داشته شود. معنایش این است که منابع مهمی در حال استفاده اند.

Database System	۱- ایجاد اتصال ۲- ایجاد پرس و جو و خواندن نتیجه از Database ۳- // ... ۴- قطع اتصال از Database	Client Service

و در نتیجه، تعداد کاربرانی که به Database میتوانند دسترسی داشته باشند و از برنامه ی Net. استفاده کنند، کاسته میشود. (به ویژه در محیطهایی که تعداد همزمان کاربران بانک اطلاعات زیاد باشد) همانطور که در شکل بالا می بینید همیشه اتصال باز است یا برقرار است. اما در ADO.Net از استدلال کاملاً متفاوتی استفاده می‌کند. وقتی در ADO.Net با بانک اطلاعات ارتباط برقرار می کنیم، اطلاعاتی که از بنک اطلاعات دریافت میکنیم، در یک Dataset قرار میگیرد. اگر اطلاعات موجود در Dataset را تغییر دهیم اتصالات موجود در جدول متناظر با بانک اطلاعات تغییر نمیکند. معنایش این است که با خیال راحت می توانید اطلاعات موجود در Dataset را دست کاری کنید. زیرا از اتصال فعال استفاده نمیشود. در صورت نیاز Dataset می تواند با منبع داده ی اصلی ارتباط برقرار کند و تمام تغییرات را اعمال کند.

ADO.Net مجموعه ای از کلاسها برای کار با داده است. فناوری ADO.Net نسبت به ADO دارای مزایایی است که برخی از آنها عبارت اند از:

۱- **قابلیت اتصال:** فناوری ADO.Net از XML به عنوان فرمت ارسال اطلاعات از یک منبع داده به مقصد مورد نظر استفاده میکند.
۲- **قابلیت نگهداری:** با افزایش کاربران مشکلاتی در رابطه با منابع موجود بروز میکند. در ADO.Net با استفاده از از برنامه ی لایه ای (N-Tire) می‌توان منطق برنامه را بین چندین لایه اضافی، توزیع نمود، چون از Cache به منظور از نگهداری از نسخه هایی از داده استفاده می شود.
۳- **قابلیت برنامه نویسی:** مدل برنامه نویسی ADO.Net کاملاً داده ی نوع قوی را پشتیبانی می کند تا کد ها شفاف تر، مختصر تر و با سادگی بیشتر نوشته شوند.
۴- **کارایی:** ADO.Net این امکان را فراهم کرده است تا عملیات اضافی مربوط به تبدیع نوع داده ها حذف شود و این امر باعث افزایش کارایی آن شده است.
۵- **توسعه پذیری:** با توجه به اینکه داده در حافظه ی محلی پنهان (Cache) نگهداری میشود، نیازی به نگهداری بلاک هایی از بانک اطلاعات یا نگهداری اتصالات فعال به بانک اطلاعات برای مقاطع زمانی بعدی، وجود نخواهد داشت، بنابراین این منبع را میتوان در اختیار برنامه های بیشتری قرار داد.

به منظور دستیابی داده در ADO.Net، ۳ لایه وجود دارد که عبارت اند از:

۱- **لایه فیزیکی محل ذخیره سازی:** می تواند بانک اطلاعاتی OLE/ODBC، Oracle، SQL Server یا فایل دیگری (XML) باشد.
۲- **لایه تامین کننده داده:** شامل اشیائی از قبیل Connection، Commandو اشیاءِ دیگری که داده را در حافظه نگهداری می کنند.
۳- **لایه‌ی Dataset:** این لایه Record های بازیابی شده از یک منبع داده را در خود نگهداری میکند. Dataset شامل رکوردهایی از یک یا چند جدول می باشد.

ADO.Net از ۲ نوع کلاس تشکیل میشود که عبارت اند از کلاس های با اتصال و کلاس های بی اتصال

۱- **کلاس های با اتصال:** این کلاسها امکانات بازیابی داده ها از منابع داده را فراهم می‌سازند و به برنامه نویس امکان اجرای فرمان جهت تغییر داده را می دهد. برای مثال کلاسهای Connection، Command، Data Reader، Data Adapter
۲- **کلاس های بی اتصال:** این کلاسها به برنامه نویس امکان دستکاری فایل بانک اطلاعاتی (یک منبع داده ی انتزاعی) را می دهد. مثل کلاس های Data Set، Data Table، Data Command، Data Row و ...

فضای نام System.Data: کلاسهای اصلی ADO.Net در فضای نام System.Data قرار دارد. این فضای نام برای دسترسی و دستکاری داده های بانک اطلاعات به کار می رود. فضای نام System.Data در فایل اسمبلی (زبان ماشین) System.data.dll قرار دارد. در فضای نام System.Data چندین فضای نام دیگر نیز وجود دارد که عبارت اند از:

۱- **فضای نام System.Data.Cmmon**: کلاسهای کاربردی واسط های مورد نیاز برای تامین کننده های داده‌ی Net. را فراهم می آورد.
۲- **فضای نام System.Data.SqlServer**: کلاسهای تامین کننده ی SqlServer را فراهم می کند. (نسخه های ۷ و بالاتر)
۳- **فضای نام System.Data.OLEDB**: تامین کننده ی داده ی OLEDB (Object Linking and Embedding Database) را فراهم می‌سازد.
۴- **فضای نام System.Data.Oracleclient**: تامین کننده ی داده ی Oracle Client را فراهم می‌کند.

تامین کننده های داده (Data Providers): اشیائی هستند که امکان برقراری ارتباط با یک منبع داده (بانک اطلاعات) را فراهم می کنند و می توانند دستوراتی را بر روی منابع داده اجرا کرده و نتایج را بازیابی کنند. تامین کننده های داده از اشیائی مانند Database Adapter، Database Connection و Database Command تشکیل شده اند. تامین کننده های داده ی مقطعی وجود دارند. مانند ODBC، SQL Server و OLEDB

کلاسهای Connection: اولین قدم برقراری ارتباط با بانک اطلاعات بین برنامه ی کاربردی و بانک اطلاعاتی می باشند. این کلاسهای علاوه بر برقراری اتصال با بانک اطلاعاتی، شروع تراکنش را امکان پذیر می سازند. کلاس Connection دارای خواصی است که در این خواص می توان نام سرویس دهنده، نام بانک اطلاعات، نام کاربر، اطلاعات احراز هویت کاربر و اطلاعات دیگری را در هنگام اتصال به بانک اطلاعاتی تعیین کرد. از آنجایی که داده ی متعددی وجود دارد بنابراین کلاسهای متفاوتی جهت اتصال به بانک اطلاعاتی ارائه می شوند که عبارت اند از:

۱- **ODBC Connection**: برای برقراری اتصال با منبع داده ی ODBC استفاده می شود.
۲- **OLEDB Connection**: برای برقراری اتصال با منبع داده ای استفاده می شود که از OLEDB پشتیبانی می کند. مثل Access
۳- **SQL Connection**: برای اتصال با بانک اطلاعاتی SQL Server مورد استفاده قرار می گیرد.
۴- **Oracle Connection**: برای اتصال با بانک اطلاعاتی Oracle بکار می رود.

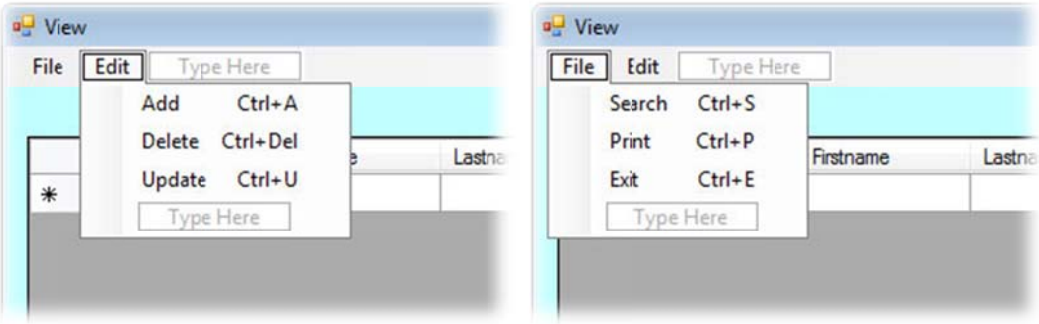
رشته اتصال (Connection String): اگر بخواهیم اطلاعات را بین برنامه کاربردی و بانک اطلاعاتی انتقال دهیم، باید یک اتصال با بانک اطلاعات برقرار کنیم. برای برقراری اتصال باید پارامتر های خاصیت رشته اتصال شیءِ Connection را تعیین کنیم.

<u>ارتباط بانک اطلاعاتی در شیءِ Data Grid View (بایند کردن با روش کد نویسی):</u>

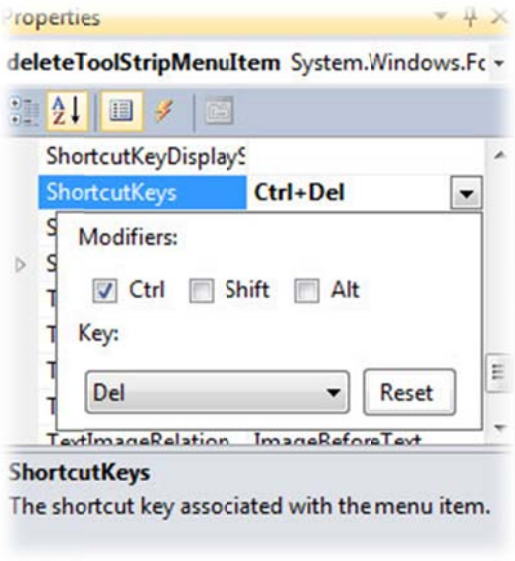
با ایجاد یک پروژه از نوع Windows Form و قرار دادن یک شیءِ Data Grid View بر روی فرم و نیز یک Button با متنِ Refresh عملیات زیر را جهت کد نویسی انجام می دهیم: بر روی Button1 (همان دکمه ی Refresh) دوبار کلیک کرده تا وارد محیط برنامه نویسی آن شویم. در این محیط کد های زیر را قرار می‌دهیم:(میتوانیم از پروژه ای که قبلاً ساختیم استفاده کنیم و اشیاءِ مذکور را در فرم Viewِ آن قرار دهیم)

قبل از کدنویسی در این محیط (فرمِ View) بایستی در قسمت بالای این صفحه ی کد نویسی (قسمت Using) دستور Using System.Data.OleDB; را اضافه کنیم که با این روش اجازه ی استفاده از دستوراتی که با OleDB شروع می‌شوند را خواهیم داشت.

در فرم ذکر شده در بالا چنان چه بخواهیم به فرم های دیگر (Add، Search و ...) دسترسی داشته باشیم، باید از یک کنترل Menu Strip استفاده کنیم. در Menu Strip ایجاد شده گزینه ی File و Edit را ایجاد کرده و برای آنها مشخصه های زیر را برای دستیابی به فرم های گوناگون اضافه می کنیم. در گزینه ی فایل عبارات Search، Print و Exit و در گزینه ی Edit، عبارات Add، Delete و Update را اضافه می کنیم:



همچنین می توان برای تمامی این عبارات یک کلید میانبر (Shortcut) ایجاد کرد. برای این کار در قسمت Propertiesِ این عبارات وارد شده و در قسمت Shortcut Keys آنها کلید های مورد نظر را به عبارات نسبت می دهیم:



سپس وارد محیط برناه نویسی این عبارات شده و آنها را جهت انتقال به فرم مورد نظر کد نویسی می کنیم. مثلا برای فرم Search از دستور زیر استفاده می کنیم:

```
Search f = new Search();
f.ShowDialog();
```

و برای گزینه ی Add نیز دستور زیر را وارد می کنیم.

```
Add f = new Add();
f.ShowDialog();
```

و همین طور برای گزینه های دیگر نیز با این ۲ خط کد عملیات ارجاع را انجام می دهیم. گزینه ی Exit که مربوط به خروج از این فرم می باشد را نیز با کد This.Close برای خروج آماده می کنیم. اما برای چاپ کردن با استفاده از پنجره ی Print مربوط به ویندوز باید از یک کنترل Print Dialog استفاده کنیم. پس این کنترل را بر روی فرم قرار می دهیم. حال بر روی گزینه ی Print نیز ۲ بار کلیک می کنیم و وارد محیط برنامه نویسی اش میشویم و دستورات زیر را وارد می کنیم.

```
PrintDialog1.showDialog();
```

با این دستور پنجره ی مربوط به Print ویندوز برای ما فعال می شود.

در ادامه ی کار وقتی روی دکمه ی Refresh در فرم View ۲بار کلیک کردیم تا وارد محیط برنامه‌نویسی اش شویم باید کد های زیر را وارد کنیم:

```
dataGridView1.DataBindings.Clear();
```

```
dataGridView1.ReadOnly = true;
string strSql, strCon;
strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
OleDbConnection Con = new OleDbConnection(strCon);
Con.Open();
strSql = "Select*From DBA";
OleDbDataAdapter da = new OleDbDataAdapter(strSql, Con);
DataSet ds = new DataSet();
da.Fill(ds, "DBA");
dataGridView1.DataBindings.Add(new Binding("datasource", ds, "DBA"));
Con.Close();
```

در خط اول این برنامه توسط دستور Clear اطلاعات قبلی Data Grid View را پاک کرده که با این روش تداخلی در نمایش اطلاعات به وجود نخواهد آمد. در خط دوم این کد نیز قابلیت فقط‌خواندنی‌بودن فیلد ها را به Data Grid View داده ایم. در خطوط سوم و چهارم دو رشته به نام های strCon و strSql تعریف کرده ایم که هر کدام از این رشته ها به ترتیب زیر به کار می‌روند.

رشته ی strCon برای ارتباط با پایگاه داده مان (جدولی که در بخش «نحوه ی ساخت جدول در Access» توضیحِ ساختن آنرا داده‌ایم) بکار می رود و رشته ی strSql برای ارتباط با جدول داخل پایگاه داده مان بکار می رود. در خط پنجم رشته اتصال strCon را مقدار دهی کرده ایم و در قسمت Data Source آن، مسیرِ پایگاه داده را وارد نموده ایم.

توجه: در هنگام مسیردهی پایگاه داده، اگر آدرس کامل فایل را ذکر می کنید، حتماً بایستی از دو بک‌اسلش (\\) استفاده کنید و همینطور نیز از پسوند mdb در انتهای نام پایگاه داده تان بایستی استفاده کنید. مثلا:
C:\\DBA.mdb

در خط ششم این کد نیز شیءِ Con را ایجاد کرده و مقدار رشته اتصال را داخل آن قرار میدهیم (برای ایجاد رشته اتصال با بانک اطلاعاتی). در خط هفتم این کد، شیءِ اتصال را باز کرده و در خط هشتمِ آن توسط دستور Select*From DBA جدول مورد نظر در پایگاه داده مان را انتخاب می کنیم. در خط نهم توسط کلاس Data Adapter شیءِ da را ساخته ایم. این کلاس برای دریافت نتایج تقاضای بانک اطلاعاتی و قرار دادن در Dataset و همچنین ذخیره تغییرات انجام شده در Dataset در بانک اطلاعاتی بکار میرود. در خط بعدی کلاس Dataset که هسته ی ADO.Net هست اطلاعات حاصل از تقاضای بانک اطلاعاتی در آن ذخیره میشود.

جهت فارسی کردن مطالبی که در این Data Grid View وجود دارد کافیست از دستور زیر جهت فارسی کردن ستون Data Grid View استفاده نماییم. دستور زیر را قبل از ;(Con.Close() بایستی اضافه کنیم:

```
dataGridView1.Columns[0].HeaderText = "ردیف";
dataGridView1.Columns[1].HeaderText = "نام";
dataGridView1.Columns[2].HeaderText = "خانوادگی نام";
dataGridView1.Columns[3].HeaderText = "بیمار کد";
dataGridView1.Columns[4].HeaderText = "آدرس";
dataGridView1.Columns[5].HeaderText = "تلفن";
dataGridView1.Columns[6].HeaderText = "ایمیل";
```

تذکر: برای کنترل اشیاء در یک فرم توسط کلید Tab کیبورد، از منوی View گزینه ی Tab Order را انتخاب کرده و در این قسمت به ترتیب اشیاء را نوبت بندی می کنیم.

طراحی فرم Search:

کنترل های زیر را در فرم Search ایجاد می کنیم: Button، Text Box، Label، Group Box، Menu Strip. Button و Text Box و Label را داخل Group Box قرار می دهیم. نام Label را به مقدار Enter Lastname تغییر نام می دهیم و نام Button را به بعبارت Search تغییر می دهیم. نام Group Box را نیز به عبارت Search info تغییر می دهیم. یک کنترل Data Grid View نیز بر روی فرم ایجاد می کنیم. در Menu Strip گزینه های مورد نظر را برای دستیابی به فرم های دیگر را مانند فرم View ایجاد کرده و کد های مربوط به هر کدام از آنها را وارد می کنیم (مانند فرم View). در ادامه یک کنترل Print Dialog نیز برای چاپ محتویات از طریق پنجره ی Print ویندوز به داخل فرم می‌آوریم.

سپس بر روی دکمه ی Search دوبار کلیک کرده و کد های زیر را وارد می کنیم.

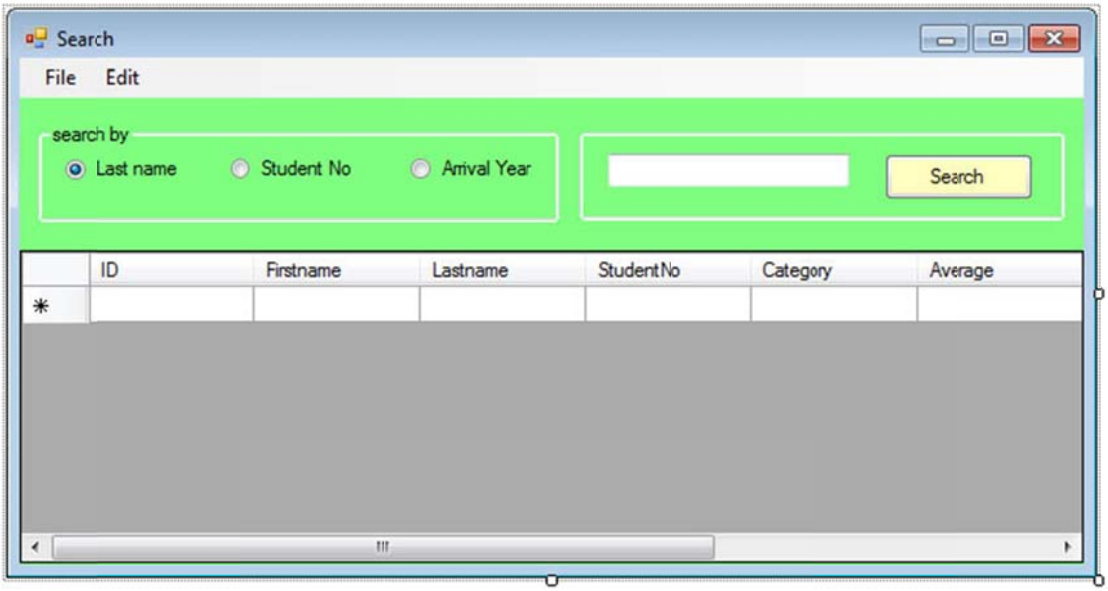
```
dataGridView1.DataBindings.Clear();
dataGridView1.ReadOnly = true;
string strSql, strCon;
strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
OleDbConnection Con = new OleDbConnection(strCon);
Con.Open();
strSql = "Select*From Hos Where Lastname=" + " ' " + textBox1.Text + " ' ";
OleDbDataAdapter da = new OleDbDataAdapter(strSql, Con);
DataSet ds = new DataSet();
da.Fill(ds, "DBA");
dataGridView1.DataBindings.Add("datasource", ds, "DBA");
```

کد های بالا به مانند کد های فرم View می باشد. اما در جایی که رشته ی اتصال به جدول پایگاه داده (strSql) تعریف شده اند مقدار آن با مقدار این رشته در فرم View متفاوت است. در این رشته با استفاده

از سه عنصر Select، From و Where جدول موردنظر (DBA) و همینطور فیلد موردنظر (Lastname) مورد جستجو قرار می گیرد. به عبارتی محتویات وارد شده در Text Box با محتویات موجود در فیلدِ Lastname جدولِ DBA مقایسه شده و عملیات Search انجام می‌شود.

تذکر: نام فیلدی که در رشته strSql مورد جستجو قرار می گیرد باید دقیقاً مانند نام فیلدی باشد که در جدولمان آن را نام گذاری کرده ایم. همان طور که می دانید در بین نام های فیلد هایی که در جدول وجود دارند فاصله ای وجود ندارد.

زمانی که کاربر بخواهد از چند فیلد متفاوت برای عملیات Search استفاده کند بایستی از اشیائی مانند Radio Button استفاده کند. در این مثال ما از سه دکمه ی رادویی به نامهای Lastname، Email و Telephone استفاده کرده ایم:



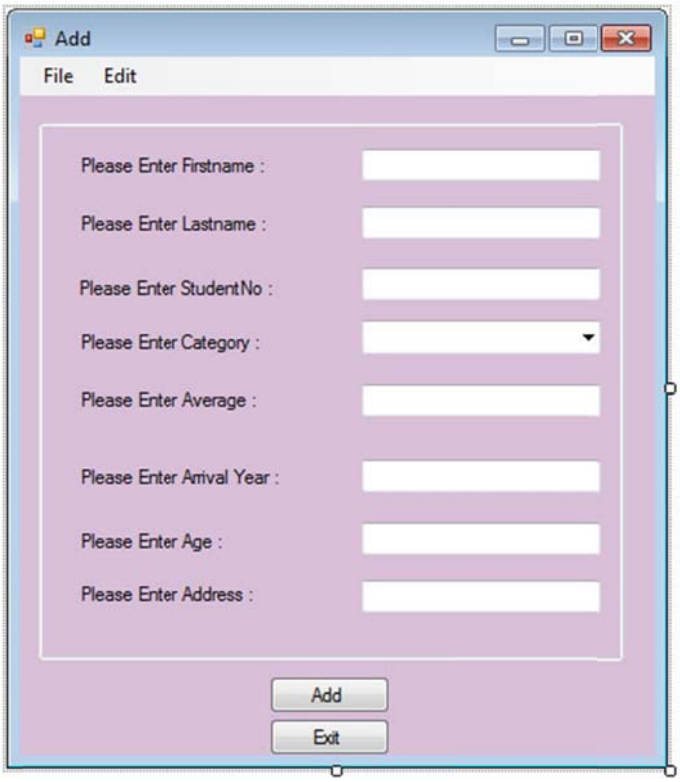
برای اینکه یکی از این دکمه های رادیویی را بتوانیم در حالت پیشفرض انتخاب کنیم در قسمت Properties آنها وارد شده و خاصیت Checked آن را به True تغییر می دهیم و برای انتخاب هر یک از این فیلد ها از یک شرط بصورت زیر استفاده می کنیم:

```
if (radioButton1.Checked)
{
   dataGridView1.DataBindings.Clear();
   dataGridView1.ReadOnly = true;
   string strSql, strCon;
   strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
   OleDbConnection Con = new OleDbConnection(strCon);
   Con.Open();
   strSql = "Select*From DBA Where Lastname=" + " ' " + textBox1.Text + " ' ";
   OleDbDataAdapter da = new OleDbDataAdapter(strSql, Con);
```

```
    DataSet ds = new DataSet();
    da.Fill(ds, "DBA");
    dataGridView1.DataBindings.Add("datasource", ds, "DBA");
}
else if (radioButton2.Checked) {
    dataGridView1.DataBindings.Clear();
    dataGridView1.ReadOnly = true;
    string strSql, strCon;
    strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
    OleDbConnection Con = new OleDbConnection(strCon);
    Con.Open();
    strSql = "Select*From DBA Where StudentNo=" + " ' " + textBox1.Text + " ' ";
    OleDbDataAdapter da = new OleDbDataAdapter(strSql, Con);
    DataSet ds = new DataSet();
    da.Fill(ds, "DBA");
    dataGridView1.DataBindings.Add("datasource", ds, "DBA");
}
else if (radioButton3.Checked) {
    dataGridView1.DataBindings.Clear();
    dataGridView1.ReadOnly = true;
    string strSql, strCon;
    strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
    OleDbConnection Con = new OleDbConnection(strCon);
    Con.Open();
    strSql = "Select*From DBA Where ArrivalYear=" + " ' " + textBox1.Text + " ' ";
    OleDbDataAdapter da = new OleDbDataAdapter(strSql, Con);
    DataSet ds = new DataSet();
    da.Fill(ds, "DBA");
    dataGridView1.DataBindings.Add("datasource", ds, "DBA");
}
```

نحوه ی ساختن فرم **Add**:

کنترل های زیر را بر روی فرمView ایجاد می کنیم: ۶ عدد Text Box و ۶ عـدد Label ایجـاد کـرده و آنها را داخل Group Box قرار می دهیم. ۲ عدد Button و یک Menu Strip نیـز ایجـاد مـی کنـیم و مقادیر Menu Strip را مانند فرم های قبلی می سازیم. نام Button ها را نیز به Add و Exit تغییر مـی دهیم. نام Group box را نیز به عبارت Insert تغییر می دهیم. نام Label ها را نیز به ترتیب به عبارات زیر تغییر می دهیم: First Name، Last Name، Code، Address، Email.



در قسمت کد نویسی این فرم وارد شده و در قسمت Using ها عبارت using System.Data.OleDb; را وارد کرده و در قسمت Public Partial Class وارد می کنیم:

```
string strSql, strCon;
OleDbDataAdapter da = new OleDbDataAdapter();
DataSet ds = new DataSet();
OleDbConnection Con = new OleDbConnection();
DataRow newRow;
OleDbCommandBuilder cb = new OleDbCommandBuilder();
```

در مورد دستور DataRow در خط پنجم کد های بالا در شیءِ DataRow سطری از اطلاعات جدول را نشان می دهد. با استفاده از نام فیلد می توان به هر مقداری از آن سطر دست یافت و متغیر newRow رکورد جدیدی را که باید در DataSet اضافه شود ذخیره می کند و در خط بعدی نیز شیءِ cb برای تولید دستورات بکار می رود.

در فرم Add بر روی دکمه ی Add دوبار کلیک کرده و وارد محیط برنامه نویسی آن می‌شویم. کد های زیر را وارد می کنیم:

```
if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" ||
textBox4.Text == "" || textBox5.Text == "" || textBox6.Text == "" ||
textBox7.Text == "" || textBox8.Text == "")
   MessageBox.Show("Please Fill!!", "Error");
else
   MessageBox.Show("Add Person", "Add");
newRow = ds.Tables["DBA"].NewRow();
newRow["Firstname"] = textBox1.Text;
newRow["Lastname"] = textBox2.Text;
newRow["StudentNo"] = textBox3.Text;
newRow["Category"] = textBox4.Text;
newRow["Average"] = textBox5.Text;
newRow["ArrivalYear"] = textBox6.Text;
newRow["Age"] = textBox7.Text;
newRow["Address"] = textBox8.Text;
ds.Tables["DBA"].Rows.Add(newRow);
cb = new OleDbCommandBuilder(da);
da.InsertCommand = cb.GetInsertCommand();
da.Update(ds, "DBA");
Con.Close();
textBox1.Text = "";
textBox2.Text = "";
textBox3.Text = "";
textBox4.Text = "";
textBox5.Text = "";
textBox6.Text = "";
textBox7.Text = "";
textBox8.Text = "";
```

در کد های مذکور برای باتنِ Add در سطر های اول این برنامه شروطی را مبنـی بـر خـالی نبـودن Text Box ها بیان کردیم و با دستور MessageBox آن را کنترل نمودیم. پس از تعریف متغیـر newRow از نوع DataRow که سطری از اطلاعات جدول را نشان می دهد، کلیـه ی فیلـد هـای موجـود در جـدول را (مانند Firstname یا Lastname) به Text Box های مربوط به آنها مقید نمودیم. سپس توسط دستور Update تثبیت تغییرات DataSet را انجام دادیم و در انتهـا Text Box هـا را خـالی نمـودیم تـا کـاربر بتواند داده های جدید را وارد کند.

تذکر: کدهایی که تا اینجا بیان شدند و از این به بعد بیان می شوند و همچنین روش های بیان شده پایـه و اساس کد نویسی برای هر نوع Databaseی می باشد. مثلاً از همین روال می توان برای کد نویسـی پایگـاه داده ی یک بیمارستان، نیروی انتظامی، نمایشگاه ماشین و ... استفاده کرد. فقط ممکن است تغییرات جزئـی در روال کد نویسی برای آن سازمان خاص انجام پذیرد. (به مثال زیر توجه کنید)

مثلا برای فرم Add اگر نیاز باشد از یک Combo Box استفاده کنیم به صورت زیر عمل می کنیم:

یک کنترل Combo Box را در فرم Add به جای Text Box مربوط بـه Category قـرار مـی دهـیم. برای پر کردن اطلاعات داخل این Combo Box از خاصـیت بایـنـد کـردن Combo Box اسـتفاده مـی کنیم. مانند روش بایند کردن جدول در Data Grid View. به عبارتی به یک جدول جدید مـثلا جـدولی به نام tableGroup نیاز داریم که در آن نام شهر های مختلف وجود داشته باشد.

در Combo Box با زدن گزینه ی Use Data Bound Items می تـوانیم جـدول مـورد نظـر را بایـنـد کنیم. در قسمت Data Source، جدول را انتخاب و در قسمت Display Member فیلـد مـورد نظـر را انتخاب می کنیم. در محیط کد نویسی بدلیل اینکه ما Text Box مورد نظر را حذف نموده ایم و بـه جـای آن از Combo Box اسـتفاده نمـوده ایـم بایـد در کـد نویسـی نیـز بـه جـای آن Text Box از دسـتور comboBox1.Text استفاده کنیم.

<u>روش طراحی و کد نویسی فرم Delete:</u>

بر روی فرم Delete کنترل های زیر را اضافه می کنیم: Button، Label، Text Box، Group Box و یک Menu Strip.

Menu Strip را مانند فرم های قبلـی (View و Search) آمـاده مـی کنـیم و گزینـه هـای آن را بـرای دسترسی به صفحات و فرم های مورد نظر کد نویسی می کنیم.

عنوان Label را به عبارت Enter Last Name تغییر می دهیم و عنوان Button را به عبـارت Delete تغییر نام می دهیم. این سه کنترل را داخل Group Box با نام Information for Delete! قرار مـی-دهیم. بر روی باتنِ Delete دو بار کلیک می کنـیم تـا وارد محـیط برنامـه نویسـی اش شـویم. در قسـمت Usingها دسـتور using System.Data.OleDb; را وارد مـی کنـیم تـا اسـتفاده از دسـتوراتی کـه بـا OleDb شروع می شوند برای ما امکان پذیر شود. در قسمت Public Partial Class نیز کد های زیـر را وارد می کنیم:

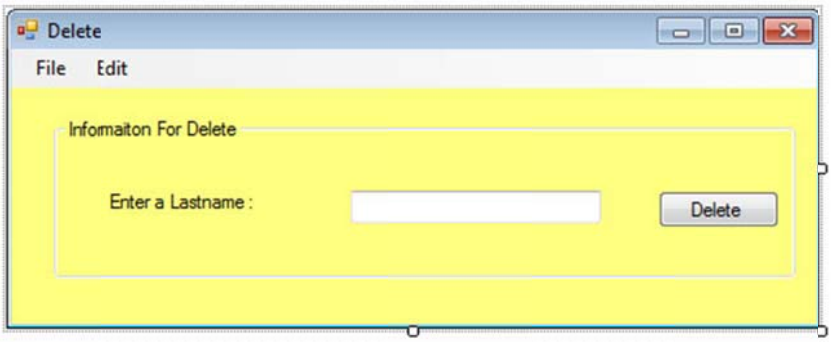
```
string strCon, strSql;
OleDbConnection Con = new OleDbConnection();
OleDbDataAdapter da = new OleDbDataAdapter();
DataSet ds = new DataSet();
```

در کد نویسی مربوط به دکمه ی Delete نیز کد های زیر را وارد می کنیم:

```
strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
```

```
Con = new OleDbConnection(strCon);
Con.Open();
strSql = "Delete From DBA Where Lastname="+" ' "+textBox1.Text+" ' ";
da = new OleDbDataAdapter(strSql,Con);
da.Fill(ds,"DBA");
ds.Clear();
textBox1.Text = "";
```

تنها فرق این کد ها با برنامه های نوشته شده در بخش Search و View در این است که در رشته اتصال به جدول پایگاه داده (strSql) تغییرات ایجاد شده است.



برخی دیگر از خواص فرم:

همان طور که در گذشته گفته شد اولین مرحله در برنامه نویسی ایجاد فرم می باشد. فرمی که دارای کنترل های مختلفی است. فرم دارای خواصی است که رفتار آنرا تعیین می کند. برخی از این خواص با همین عملکرد در عناصر دیگر تکرار می شوند. برخی از خواص مهم فرم عبارت اند از:

- ***Name***: برای تعیین نام کنترل (فرم) بکار می رود. نام فرم از قانون نامگذاری متغیر ها و کنترل ها پیروی می کند. بعنوان مثال نام اولین فرمی که در یک پروژه ساخته می شود، Form1 است که با استفاده از خاصیت Name می توان آنرا تغییر داد. این خاصیت در داخل پرانتز قرار دارد تا در ابتدای لیست خواص قرار گیرد.
- ***Allow Drop***: مشخص می کند که آیا داده ای که کاربر با ماوس به فرم ارسال می کند توسط فرم پذیرفته شود یا خیر.
- ***Auto Scroll***: تعیین می کند که آیا نوار جابجایی (Scroll Bar) مورد نظر به فرم اضافه شود یا خیر. اگر مقدار این خاصیت به True تغییر کند نوار جابجایی به فرم اضافه می شود.
- ***Cancel Button***: دکمه ای را تعیین می کند که اگر کاربر کلید Esc بر روی کیبورد را فشار دهد دستوراتِ مربوط به رویدادِ کلیکِ آن دکمه اجرا شود.

- ***Control Box***: تعیین می کند آیا در فرم فرم دکمه های بیشینه (Maximize) یا کمینه (Minimize) یا بستن (Close) ظاهر شوند یا خیر.
- ***Auto Validate***: وقتی فوکوس کنترل هایی که روی فرم قرار دارند تغییر کند بطور خودکار اعتبار سنجی شوند یا خیر.
- ***Enabled***: تعیین می کند آیا فرم فعال باشد یا خیر.

برخی از رویداد های فرم: همان طور که می دانید فرم دارای رویداد هایی است که در حالات خاصی رخ می دهند. بطور مثال:

- ***رویداد Click***: وقتی رخ می دهد که کنترل کلیک شود.
- ***رویداد Key Down***: وقتی رخ می دهد که کلیدی از صفحه ی کلید (Keyboard) فشرده شود.
- ***رویداد Key Up***: این رویداد وقتی رخ می دهد که کلید فشرده شده، رها شود.

برای فرم Search می توان جستجو را بر اساس یافتن رکورد مورد نظر نیز انجام داد. برای اینکار بر روی دکمه ی Search دوبار کلیک کرده، در محیط برنامه نویسی آن کد های زیر را وارد کنید. باید توجه داشت که تمامی داده های جدول در Data Grid View مربوط فرم Search از پیش بایند شده باشند.

```
int index = -1;
if (rdoLastName.Checked)
   index = tableDbaBindingSource.Find("Lastname", textBox1.Text);
else if (rdoID.Checked)
   index = tableDbaBindingSource.Find("StudentNo", textBox1.Text);
else if (rdoArrival.Checked)
   index = tableDbaBindingSource.Find("ArrivalYear", textBox1.Text);

if (index > -1) // find record
   tableDbaBindingSource.Position = index;
else
   MessageBox.Show("رکورد مورد نظر موجود نمی باشد");
```

Binding Source ها کلاس هایی هستند که برای Bind شدنِ بین جداول Data Set و اشیائی مانند Data Grid View استفاده می شوند. که این Binding Source ها در داخل خود دارای خاصیتِ مدیریتی می باشند که از آن برای مدیریت Bind کردن استفاده می کنیم.

در سطر اول این کد، متغیری با مقدار اولیه ی صفر تعریف نموده ایم تا بعدا در حین روال کد نویسی بتوانیم به آن مقدار دهی کنیم. در خط های دوم تا هفتم سه شرط مربوط به انتخاب شدن دکمه های رادیویی را ذکر نموده ایم و در این خطوط با استفاده از دستور Find و مقید کردن Text Box به فیلد مورد نظر، عملیات جستجو را انجام دادیم. در صورتی که هر کدام از مقادیر این شرط ها درست باشد، یک مقداری را در متغیر index قرار می دهد. پس در خط هشتم شرط صفر نبودن متغیر index اجرا میشود و مقدار آن در خط نهم این برنامه توسط دستور Position حرکت بین رکورد ها را انجام می دهد.

اگر هیچ کدام از شرط های مربوط به دکمه های رادیویی اجرا نشود و یا اگر مقدار وارد شده در Text Box در بانک اطلاعاتی موجود نباشد اررورِ مربوطه در خط یازدهمِ این کد نمایش داده میشود.

تمرین۱: چنان چه چند نام فامیل یکسان (مثلا فامیل اکبری) وجود داشته باشد می‌خواهیم تمام آنها نمایش داده شود. (از روش جستجوی بین رکورد ها: روش دوم که در بالا ذکر شد)

تمرین ۲: چنان چه بر روی مشخصات فرد جستجو شده دوبار کلیک کنیم فرم جدیدی باز شده و مشخصات کامل تری از این فرد را به ما نمایش دهد. (بعنوان مثال در فرم باز شده فیلد های کد ملی و نوع بیماری و شماره اتاق و ... نمایش داده شود)

تمرین ۳: در فرم Search وقتی حرکت بین رکورد ها انجام شد، کل سطر مربوطه، آبی شود.

تمرین ۴: فرم Login را طوری بنویسید که اگر ۱۰۰ یوزرِ دیگر نیز بخواهند عضو شوند، امکان پذیر باشد.

روش دوم برای طراحی فرم Delete:

در این روش فرم Delete فقط دارای سه کنترلِ Data Grid View، Button و Menu Strip می باشد. پس از اینکه در قسمتِ Using ها دستور using System.Data.OleDb; را نوشتیم در قسمت Public Partial Class کد های زیر را وارد می کنیم:

```
string strCon, strSql;
OleDbConnection Con = new OleDbConnection();
OleDbDataAdapter da = new OleDbDataAdapter();
DataSet ds = new DataSet();
string seletedRow();
```

در چهار خط اول این کد مانند گذشته اشیاءِ مربوطه را تعریف می کنیم. اما رشته ای بنام SelectedRow نیز تعریف نموده ایم که بعداً آن را در قسمت رویداد های مربوط به کنترل Data Grid View مقدار دهی می کنیم تا با کلیک بر روی رکورد مورد نظر آن شخص انتخاب شده و ما بتوانیم عملیات Delete را انجام دهیم. در ادامه در قسمت کد نویسی باتن Delete وارد شده کد های زیر را برای آن وارد می کنیم:

```
strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
Con = new OleDbConnection(strCon);
Con.Open();
```

```
strSql = "Delete From DBA Where ID=" + "" + selectedRow + "";
da = new OleDbDataAdapter(strSql, Con);
da.Fill(ds, "DBA");
ds.Clear();
MessageBox.Show("deleted");
// TODO: This line of code loads data into the 'dBADataSet.DBA' table. You can move, or remove it, as needed.
this.dBATableAdapter.Fill(this.dBADataSet.DBA);
// TODO: This line of code loads data into the 'dBADataSet.tableGrop' table. You can move, or remove it, as needed.
strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
strSql = "Select*from DBA";
Con = new OleDbConnection(strCon);
Con.Open();
da = new OleDbDataAdapter(strSql, Con);
da.Fill(ds, "DBA");
```

در ابتدا جدول مورد نظر را در Data Grid View فرمِ Delete بایند می کنیم.

نکته: هر گاه جدول را در Data Grid View فرم خاصی بایند کنیم بر روی فرم دو بار کلیک کنیم و وارد قسمتِ برنامه نویسی آن شویم (یعنی در قسمت Form_Load) دو خط دستور بصورت خودکار در این قسمت مشاهده می کنیم که خط اول آن برای توضیح دستور خد دوم می باشد. در خط اول توضیح می دهد که دستور خط دوم مربوط به بارگذاری داده ها در Data Set جدول مربوطه می باشد. در خط دوم با استفاده از دستور Table Adapter و دستور Fill عملیات Load (بارگذاری) داده را انجام می دهد.

Table Adapter: همان طور که می دانید بطور کلی عملیات روی داده ها از چهار حالتِ Select، Insert، Update و Delete خارج نیست. برای اینکه بتوانیم از اینها استفاده کنیم باید از شیئی بنام Table Adapter استفاده کنیم. ADO.Net شرکتِ مایکروسافت شیئی جدید به نام Table Adapter را بکار میگیرد تا فرآیند مقید سازی داده ها به فرم را آسان کند. یکی از کارهایی که Table Adapter انجام می دهد اینست که شیءِ Connection، از نوعِ مناسب منبع داده ها را بکار میگیرد.

خط هشت و نهِ این کد برای نگه داشتن این اطلاعات در Data Connection در فرم مورد نظر بکار می رود.

سپس در قسمت Form_Load (با دو بار کلیک بر بروی فرم) وارد شده و کد های زیر را وارد می کنیم:

```
strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
strSql = "Select*from DBA";
Con = new OleDbConnection(strCon);
Con.Open();
da = new OleDbDataAdapter(strSql, Con);
da.Fill(ds, "DBA");
```

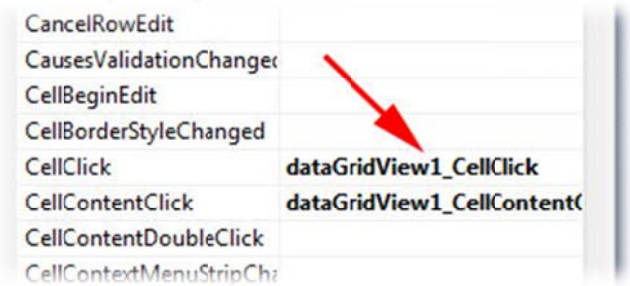
این کد ها برای این است که زمانی که فرم Delete، Load میشود کلیه ی اطلاعات جدول در ابتدای کـار قابل رویت باشند.

در فرم Delete بر روی Data Grid View دو بار کلیک کرده و وارد محیط برنامه نویسـی اش میشـویم. کد های زیر را وارد می کنیم:

```
selectedRow = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
```

در این خط از کد، متدِ ToString محتویات مربوطه را به رشته تبدیل می کند.

این خط از کد برای کلیک بر روی سـطری از Data Grid View بکـار مـی رود. توسـط دسـتور Value مقدار محتویات سطر مورد نظر در Data Grid View انتخاب میشود.



در Properties مربوط بـه Data Grid View و در قسـمت Event، مقـدار رویـداد Cell Click را برابـرِ dataGridView1_CellClick قرار می دهیم و بـا ۲ کلیلک بر روی آن کد های زیر را وارد می کنیم:

```
selectedRow = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
```

روش طراحی و کدنویسی فرم Update:

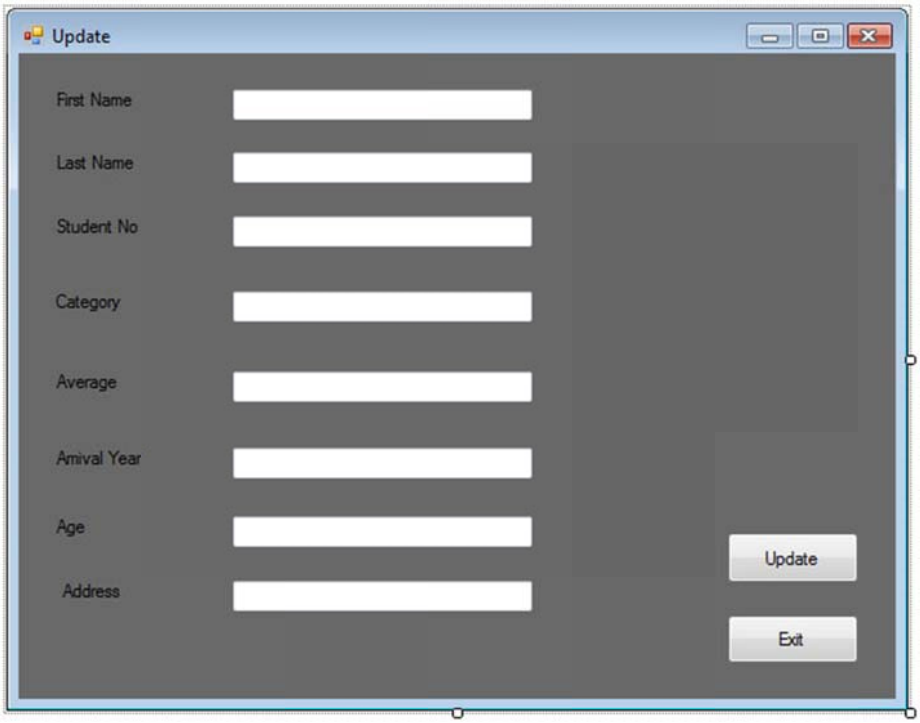
در فرم View در قسمت Menu Strip بر روی گزینه ی Update دوبار کلیک می کنیم و کد هـای زیـر را وارد می کنیم:

```
Update frmUpdate = new Update();
frmUpdate.userId = dataGridView1.CurrentRow.Cells[0].Value.ToString();
frmUpdate.ShowDialog();
```

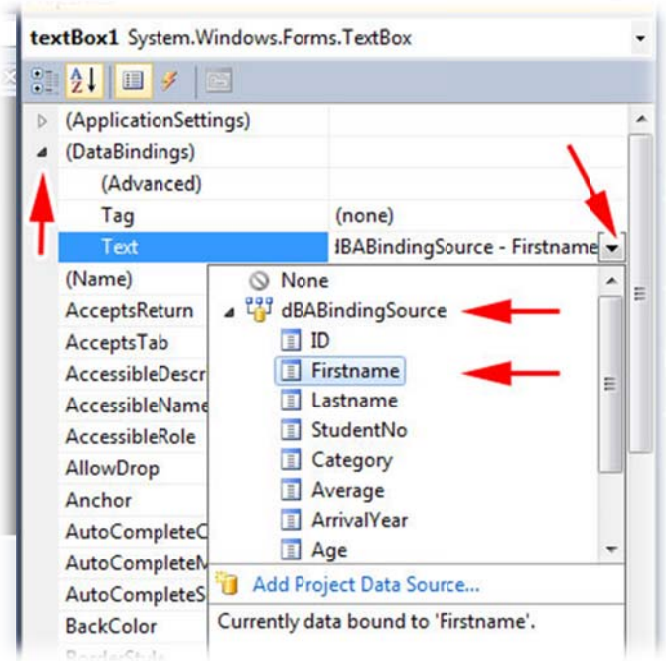
توضیح: رشته ی userId در بخش Public Partial Class مربوط به فرمِ Update بصورت زیـر تعریـف گشته است:

```
public string userId;
```

userID یک رشته بصورت عمومی می باشد و برنامه ای که در خط دوم نوشته شده برای در اختیار گـرفتن سطر های یک Data Grid View می باشد. در ادامه فرم Update را بصورت زیر طراحی می کنیم:



۸ عدد Text Box و ۸ Label مقابل آن تشکیل می دهیم؛ نام Label ها را به ترتیب برابر Firstname، Lastname، Student Number، Category، Average، Arrival Year، Age و Address وارد کرده. دو Button به نام هـای Update و Exit نیـز قـرار مـی دهـیم. از خاصـیت Data Bindings در قسمتِ Properties مربوط به Textbox ها استفاده کرده فیلد مورد نظر جدولمان را بـه Text Box هـا مقید می کنیم. (منظور خاصیت Text در Data Bindings است. مطابق شکل زیر)



در قسمت کد نویسی فرم Update وارد شده و پس از اینکه در قسمت Using ها دستور مربوط به OleDb را نوشتیم حال در قسمت Public Partial Class رشته ی userId را بصورت عمومی (Public) تعریف می کنیم.

در بخش کد نویسی Form_Load وارد میشویم کد های زیر را وارد می کنیم:

```
string strSql, strCon;
strCon = "Provider=Microsoft.jet.OLEDB.4.0;" + "Data Source=DBA.mdb";
OleDbConnection Con = new OleDbConnection(strCon);
Con.Open();
strSql = "Select * From DBA where id = " + userId;
OleDbDataAdapter da = new OleDbDataAdapter(strSql, Con);
da.Fill(dBADataSet, "DBA");
```

سپس بر روی دکمه ی Update دوبار کلیک کرده و کد های زیر را وارد می کنیم:

```
dBABindingSource.EndEdit();
this.dBATableAdapter.Update(this.dBADataSet);
this.Close();
```

در خط اول این کد توسط Binding Source ها برای ارتباط بین جدول DataSet و اشیائی مانند Data Grid View استفاده میشود را بکار گرفتیم. توسط رویداد EndEdit مقدار درون سلول را می‌توان گرفت. یعنی فقط زمانی می شود مقدار داخل سلول را در اختیار گرفت که سلول EndEdit شود.

برای ساختن فایلِ اجرایی برنامه نوشته شده در C# در قسمت Build و در قسمت Publish Windows Form Application وارد شده و برنامه را به حالت EXE تبدیل می کنیم.