نام کتاب	لینک خرید چاپی	لینک خرید الکترونیکی	لینک فایل نمونه
مبانی رایانه و برنامه‌نویسی به زبان C++	http://daneshnegar.com/book_۳۸۰۲۳۸.html	http://ktbr.ir/b۳۰۵۸۸	http://ketabesabz.com/dl/۵۲۳۱۹
آشنایی با مبانی امنیت شبکه (امنیت اطلاعات)	http://daneshnegar.com/book_۳۷۱۱۳۷.html	http://ktbr.ir/b۳۰۳۲۷	
اصول طراحی پایگاه داده‌ها	http://daneshnegar.com/book_۳۷۱۶۵۵.html	http://ktbr.ir/b۲۹۹۴۳	http://ketabesabz.com/dl/۵۲۱۵۵
آموزش گام‌به‌گام برنامه‌نویسی پایتون	http://daneshnegar.com/book_۳۹۲۵۸۲.html	http://ktbr.ir/b۲۹۹۸٤	http://ketabesabz.com/dl/۵۲۱۸۱
آزمایشگاه کامپیوتر C++ (حل مسائل مرجع کامل)	http://daneshnegar.com/book_۳۹۲۲٦۲.html	http://ktbr.ir/b۲۹۹۸۲	
آموزش گام‌به‌گام C # با LINQ	http://daneshnegar.com/book_۳٦۹۳۸۸.html	http://ktbr.ir/b۲۸٤۵۱	
C++ ساختمان داده‌ها با	http://daneshnegar.com/book_۳۷۹۰۹٤.html	http://ktbr.ir/b۲۹٦۷٦	http://ketabesabz.com/dl/۵۲۱۵٦
طراحی سیستم‌های شی‌گرا با زبان C#	http://daneshnegar.com/book_۳۷٤٦۵۸.html	http://ktbr.ir/b۲۹٦۲۱	http://ketabesabz.com/dl/۵۲۱۲٦
مدیریت استراتژیک فناوری اطلاعات	http://daneshnegar.com/book_۳۷٤٦۵۹.html	http://ktbr.ir/b۲۹۷۷۹	http://ketabesabz.com/dl/۵۲۱۲۵
گرافیک رایانه‌ای با زبان برنامه‌نویسی C#	http://daneshnegar.com/book_۳۷٦۰۲۱.html	http://ktbr.ir/b۲۹٦۷٤	http://ketabesabz.com/dl/۵۱۰٤۹
درس و کنکور پایگاه داده پیشرفته	http://daneshnegar.com/book_۳۹۲۵۷۸.html	http://ktbr.ir/b۲۹٦٤٤	http://ketabesabz.com/dl/۵۲۱۰۲
فیزیک الکتریسیته	http://daneshnegar.com/book_۳۷۹۱٦۱.html	http://ktbr.ir/b۲۹٦۸۰	
تجارت الکترونیکی	http://daneshnegar.com/book_۳۷۹۱۸۸.html	http://ktbr.ir/b۲۹٦۱۹	
راهنمای کاربردی کاربری OPNET برای شبکه‌های شبیه‌سازی کامپیوتر	http://daneshnegar.com/book_۳۹۲۵۸۳.html	http://ktbr.ir/b۲۸۵۰٤	
درس و کنکور سیستم عامل پیشرفته	http://daneshnegar.com/book_۳۹۲۵۸۰.html	http://ktbr.ir/b۲۸۵۰۵	http://ketabesabz.com/dl/۵۲۱۸۰
شبکه‌های کامپیوتری با رویکرد کاربردی، آزمایشگاه شبیه‌سازی شبکه	http://daneshnegar.com/book_۳۹۲۲۵٤.html	http://ktbr.ir/b۲۸۵۲۸	
آزمایشگاه پایگاه داده با SQL Server ۲۰۱۲	http://daneshnegar.com/book_۳۷۷۳۰۱.html	http://ktbr.ir/b۲۸۵۰۳	http://ketabesabz.com/dl/۵۳٤٤۷

کاربرد رایانه در مدیریت و حسابداری	http://daneshnegar.com/book_۳۷۵۸۹۲.html	http://ktbr.ir/b۲۸٤٥۰	
آموزش گام‌به‌گام برنامه‌نویسی بانک اطلاعاتی با ویژوال بیسیک‌نت	http://daneshnegar.com/book_۳٦۸۹۲۹.html	http://ktbr.ir/b۲۸٤٤۹	
آموزش گام‌به‌گام برنامه‌نویسی به زبان C++	http://daneshnegar.com/book_۳۸۰۲۳۸.html	http://ktbr.ir/b۲۸٤٥۲	
دانلود کتاب آموزش گام‌به‌گام برنامه‌نویسی بانک اطلاعاتی با #C	http://daneshnegar.com/book_۳٦۸٤۸٦.html	http://ktbr.ir/b۲۸٤٤۸	http://ketabesabz.com/dl/٥۱۰٤۷
حل مسائل پاسکال		http://ktbr.ir/b۲۸۳۹۸	
حل مسائل C++	http://daneshnegar.com/book_۳۹۲۲٦۲.html	http://ktbr.ir/b۲۸٤۰۱	http://ketabesabz.com/dl/٥۱۰٤۸
دانلود کتاب حل مسائل #C	http://daneshnegar.com/book_۳۷٤٦٥۷.html	http://ktbr.ir/b۲۸۳۹۹	http://ketabesabz.com/dl/٥۱۰۱۱
دانلود کتاب حل مسائل C		http://ktbr.ir/b۲۸۳۹۷	

آموزش گام به گام

LINQ با C#

(مرجع کامل)

تالیف : مهندس رمضان عباس نژاد ورزی

برخی از عناوین مهم

- آشنایی با زبان C#
- بانک اطلاعاتی SQL Server
- مفاهیم اولیه و عملگرهای LINQ
- مدل شی LINQ to SQL
- مدل شی LINQ to Dataset
- مدل شی XML to LINQ
- تراکنش‌ها و برخوردها
- گزارش‌گیری با Microsoft Report

آموزش گام به گام
LINQ با C#
(مرجع کامل)

تالیف

مهندس رمضان عباس‌نژادورزی

فن‌آوری نوین

سرشناسه	: عباس‌نژاد ورزی، رمضان، ۱۳٤۸-
عنوان و نام پدیدآور	: آموزش گام به گام LINQ با #C (مرجع کامل)/تالیف رمضان عباس‌نژاد ورزی
مشخصات نشر	: بابل:فن‌آوری نوین، ۱۳۸۹ .
مشخصات ظاهری	: ۲٤۸ ص.:مصور، جدول.
شابک	: ۶۵۰۰۰ریال:٤-٦-۹۱٤۱۳-٦۰۰-۹۷۸
وضعیت فهرست نویسی	: فیپا
یادداشت کتابنامه	: ص.۲٤۸
موضوع	: ال. آی. ان. کیو مایکروسافت
موضوع	: اس. کیو. ال (زبان برنامه نویسی کامپیوتر)
موضوع	: زبان‌های پرس و جوی کامپیوتری
موضوع	: سی شارپ (زبان برنامه‌نویسی کامپیوتر)
رده بندی کنگره	: QA۷۶/۷۳/الف ۷۳ ع ۲ ۱۳۸۹
رده بندی دیویی	۰۰۵/۲۶۸
شماره کتابشناسی ملی	۲۰۰۹۰۰۲

فن‌آوری نوین www.fanavarinovin.net
بابل، کدپستی ٤۷۱٦۷-۷۳٤٤۸ تلفن: ۲۲۵٦٦۸۷-۰۱۱۱

آموزش گام به گام LINQ با #C (مرجع کامل)

تألیف: مهندس رمضان عباس‌نژادورزی

نوبت چاپ: چاپ اول

سال چاپ: بهار ۱۳۸۹

شمارگان:۱۰۰۰ جلد

قیمت: ٦۵۰۰ تومان

نام چاپخانه و صحافی: فرنگاررنگ

شابک: ٤- ٦- ۹۱٤۱۳ - ٦۰۰ - ۹۷۸

نشانی ناشر: بابل چهارراه نواب، کاظم‌بیگی، جنب حسینیه منصور کاظم‌بیگی، طبقه همکف

طراح جلد: کانون آگهی و تبلیغات آبان (احمد فرجی)

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱٤۲ تلفکس: ٦٦٤۰۰۲۲۰-٦٦٤۰۰۱٤٤

فهرست مطالب

فصل اول : آشنایی با زبان #C

۱-۱. زبان #C

۱-۲. فضای نام

۱-۳. انواع داده‌ها

۱-٤. متغیرها

۱-٤-۱. نامگذاری متغیرها

۱-٤-۲. اعلان متغیرها

۱-٤-۳. مقدار دادن به متغیرها

۱-٥. ثوابت

۱-٦. عملگرها

۱-۷. فرم برنامه

۱-۷-۱. خواص فرم

۱-۷-۲. رویدادهای فرم

۱-۷-۳. متدهای فرم

۱-۸. کنترل‌ها

۱-۸-۱. کنترل Label

۱-۸-۲. کنترل TextBox

۱-۸-۳. کنترل Button

۱-۸-٤. کنترل ListBox

۱-۸-٥. کنترل ComboBox

۱-۸-٦. کنترل CheckBox

۱-۸-۷. کنترل CheckedListBox

۱-۸-۸. کنترل RadioButton

۱-۸-۹. کنترل GroupBox

۱-۸-۱۰. کنترل MenuStrip

۱-۸-۱۱. کنترل ContextMenuStrip

۱-۸-۱۲. کنترل PictureBox

۱-۹. ساختارهای کنترلی

۱-۹-۱. ساختارهای تصمیم

۱-۱۰. ساختارهای تکرار

۱-۱۱. مدیریت صفحه کلی

۱-۱۲. آرایه‌ها

۱-۱۳. کلاس‌ها و اشیاء

۱-۱۳-۱. تعریف کلاس

۱-۱۳-۲. نمونه‌سازی کلاس

۱-۱۳-۳. اعضای کلاس

فصل دوم: مبانی بانک اطلاعات SQL Server

۲-۱. تعریف سیستم مدیریت بانک اطلاعات

۲-۱-۱. دلایل استفاده از بانک اطلاعات

۲-۱-۲. طراحی بانک اطلاعاتی

۲-۱-۳. نرمال‌سازی داده‌ها

۲-۲. بانک اطلاعات SQL Server

۲-۳. معرفی بانک اطلاعاتی نمونه

۲-٤. ورود به بانک اطلاعاتی SQL Server

۲-٥. تایپ و اجرای دستورات SQL

۲-٦. ایجاد بانک اطلاعاتی

۲-٦-۱. تغییر خواص اطلاعاتی موجود

۲-٦-۲. حذف بانک اطلاعاتی موجود

۲-۷. اشیای بانک اطلاعات
۲-۷-۱. ایجاد جدول با دستور SQL
۲-۷-۲. تغییر ساختار جدول با دستور SQL
۲-۷-۳. حذف جدول با دستور SQL .
۲-۸. دستورات SQL برای ورود، ویرایش و حذف داده‌ها
۲-۸-۱. دستور INSERT
۲-۸-۲. ویرایش رکوردهای جدول
۲-۸-۳. حذف رکوردهای جدول
۲-۹. دستور SELECT
فصل سوم : مفاهیم اولیه و عملگرهای LINQ
۳-۱. مزایای LINQ
۳-۲. معماری LINQ
۳ - ۳ . اسمبلی‌های میانی هسته LINQ
۳-٤ . LINQ چطور پیاده‌سازی می‌شود؟
۳-۵ . به کارگیری عملگرهای پرس‌وجوی استاندارد
۳-٦ . تعریف متغیر برای نگهداری نتیجه پرس‌وجوی LINQ
۳-۷ . ایجاد پرس‌وجو از منابع داده
۳-۷-۱ . ساختار پرس‌وجوی LINQ
۳-۷-۲ . استفاده از متدهای پرس‌وجو و عبارات Lambda
۳-۸ . عملگرهای استاندارد پرس‌وجو
فصل چهارم : مدل شیء LINQ to SQL
٤ – ۱. کلاس Data Context
٤ – ۱ –۱. نوع قوی DataContext
٤ – ۱ –۲. نگاشت جداول
٤ – ۱ – ۳. نگاشت ستون‌ها
٤ – ۱ – ٤. نگاشت رابطه‌ها
٤ – ۲. دستکاری داده‌ها
٤ – ۲ – ۱. اضافه کردن رکورد
٤ – ۲ – ۲. حذف رکورد
٤ – ۲ – ۳ . به روز رسانی رکورد
٤ – ۳. کنترل DataGridView
فصل پنجم : قطعه LINQ to Dataset
۵-۱. دستیابی به بانک اطلاعات با ADO.NET
۵-۱-۱. کلاس Connection
۵-۱-۲. کلاس Command
۵-۱-۳. کلاس Dataset
۵-۱-٤. کلاس DataAdapter
۵-۱-۵. کلاس DataTable
۵-۱-٦. کلاس DataColumn
۵-۱-۷. کلاس DataRow
۵-۱-۸. کلاس DataReader
۵-۲. مروری بر LINQ to Dataset
۵-۳. نیازمندی‌های پروژه LINQ to Dataset
فصل ششم : رویه‌های ذخیره شده و توابع
٦-۱. متغیرها
٦-۲. توضیحات
٦-۳. ساختارهای تصمیم
٦-۳-۱. دستور IF … Else
٦-۳-۲. دستور IF تو در تو
٦-۳-۳. دستور CASE
٦-٤. رویه‌های ذخیره شده
٦-٤-۱. ایجاد رویه‌های ذخیره شده
٦-٤-۲. اجرای رویه ذخیره شده
٦-٤-۳. تغییر رویه ذخیره شده

٤-٤-٦. حذف رویه‌های ذخیره شده
٥-٦. توابع
١-٥-٦. ایجاد توابع
٢-٦-٦. تغییر تابع
٣-٥-٦. حذف تابع
٦-٦. نگاشت توابع و رویه‌ها ذخیره شده
١-٦-٦. نگاشت یک شیء در LINQ به یک رویه ذخیره شده با تابع در بانک اطلاعات
٢-٦-٦. معرفی پارامترها و رویه ذخیره شده LINQ
٣-٦-٦. تعریف دستوراتی که رویه ذخیره شده را اجرا می‌کنند
٤-٦-٦. برگشت مقادیر
٥-٦-٦. نگاشت رویه‌های ذخیره شده برای نتایج چندگانه
٦-٦-٦. نگاشت و فراخوانی توابع تعریف شده توسط کاربر

فصل هفتم : LINQ وXML

٧-١. ساختار داده‌های XML
٧-٢. صفات
٧-٣. بخش‌های تعریف XML
٧-٤. کلاس‌های LINQ to XML
١-٤-٧. کلاس XElement
٢-٤-٧. کلاس XAttribute
٣-٤-٧. کلاس XDocument
٧-٥. مفاهیم برنامه‌نویسی LINQ to XML
١-٥-٧. بار کردن سند XML موجود
٢-٥-٧. ذخیره کردن XML از طریق LINQ to XML
٣-٥-٧. دستکاری XML
٤-٥-٧. اضافه کردن عناصر
٥-٥-٧. به روز رسانی عناصر
٦-٥-٧. حذف عناصر
٧-٦. کار کردن با صفات
١-٦-٧. اضافه کردن صفت
٢-٦-٧. بازیابی صفت
٣-٦-٧. حذف صفت
٧-٧. رویدادهای LINQ to XML
٧-٨. رویدادهای SQL to XML

فصل هشتم : تراکنش‌ها و برخوردها

٨-١. ارجاع به تراکنش
٨-٢. همزمانی خوش‌بینانه
١-٢-٨. کلاس ChangeConflictException
٢-٢-٨. متد Reslove
٨-٣. روش‌های پیاده‌سازی تراکنش
١-٣-٨. خاصیت Transaction شیء DataContext
٢-٣-٨. کلاس TransactionScope
٣-٣-٨. کلاس CommittableTransaction

فصل نهم: گزارش گیری با Microsoft Report

٩-١. امکانات نرم‌افزار Microsoft Report
٩-٢. مراحل طراحی گزارش
٩-٣. ایجاد گزارش با ویزارد
٩-٤. اضافه کردن گزارش جدید
٩-٤-١. بخش‌های گزارش
٩-٤-٢. اضافه کردن فیلد متن به گزارش

۳-٤-۹. اضافه کردن خط
٤-٤-۹. اضافه کردن مستطیل
٥-٤-۹. اضافه کردن تصویر به گزارش
٥-۹. نمایش پنجره DataSources
٦-۹. ارسال پارامتر به گزارش
پیوست الف: امکانات طراحی LINQ
پیوست ب: ابزار SQLMetal
منابع :

مقدمه

امروزه کد و داده دو عنصر اصلی نرم‌افزار هستند. یکی از بخش‌های بسیار مهم نرم‌افزار، نوشتن کدهایی برای دستیابی به داده است. از آنجائی که منابع داده‌ای مختلف از قبیل بانک‌های اطلاعاتی متعدد، فایل‌های XML و آرایه‌ها وجود دارند، یکی از مهم‌ترین مشکلات برنامه‌نویسان، این است که هر یک از منابع داده از یک مدل خاص استفاده می‌کند. بنابراین، برنامه‌نویس باید داده‌های خود را از مدلی به مدل دیگر تبدیل کند تا بتواند از آن‌ها استفاده نماید. این تبدیلات نه تنها زمان‌بر است، بلکه برنامه‌نویس باید کار کردن با ابزارهای مختلف را یاد بگیرد. به همین دلیل، شرکت مایکروسافت تکنولوژی LINQ (Language Integrated Query) را در دات‌نت اضافه نموده است. به زبان ساده می‌توان گفت LINQ، واسطی است که بین زبان برنامه‌نویسی و منبع داده قرار می‌گیرد و ارتباط بین آن‌ها را برقرار می‌کند. یعنی، نیازی نیست با تغییر منبع داده، برنامه نوشته شده در زبان برنامه‌نویسی تغییر یابد.

در این کتاب، مقدمه‌ای از C#، بانک اطلاعاتی SQL Server، عملگرهای استاندارد LINQ، قطعات LINQ to SQL و LINQ to Dataset آمده است. همچنین، این کتاب به موضوعاتی از قبیل چگونگی نگاشت رویه‌های ذخیره شده و توابع به LINQ، ارتباط بین LINQ و XML، چگونگی پیاده‌سازی تراکنش در LINQ و ارتباط بین Microsoft Report و LINQ پرداخته است. در پیوست امکانات طراحی کلاس در LINQ و ابزار SqlMetal بیان گردیده است.

در پایان امیدوارم این اثر نیز مورد توجه اساتید و دانشجویان عزیز قرار گیرد.

کتاب حاوی برنامه‌هایی است که کد آن‌ها را می‌توانید به صورت رایگان از سایت انتشارات فن‌آوری نوین به آدرس www.fanavarinovin.net بگیرید.

عباس نژادورزی

fanavarienovin@gmail.com

نام کتاب	لینک خرید چاپی	لینک خرید الکترونیکی	لینک فایل نمونه
مبانی رایانه و برنامه‌نویسی به زبان C++	http://daneshnegar.com/book_۳۸۰۲۳۸.html	http://ktbr.ir/b۳۰۵۸۸	http://ketabesabz.com/dl/۵۲۳۱۹
آشنایی با مبانی امنیت شبکه (امنیت اطلاعات)	http://daneshnegar.com/book_۳۷۱۱۳۷.html	http://ktbr.ir/b۳۰۳۲۷	
اصول طراحی پایگاه داده‌ها	http://daneshnegar.com/book_۳۷۱۶۵۵.html	http://ktbr.ir/b۲۹۹۴۳	http://ketabesabz.com/dl/۵۲۱۵۵
آموزش گام‌به‌گام برنامه‌نویسی پایتون	http://daneshnegar.com/book_۳۹۲۵۸۲.html	http://ktbr.ir/b۲۹۹۸۴	http://ketabesabz.com/dl/۵۲۱۸۱
آزمایشگاه C++ (حل مسائل کامپیوتر مرجع کامل)	http://daneshnegar.com/book_۳۹۲۲۶۲.html	http://ktbr.ir/b۲۹۹۸۲	
آموزش گام‌به‌گام C # با LINQ	http://daneshnegar.com/book_۳۶۹۳۸۸.html	http://ktbr.ir/b۲۸۴۵۱	
C++ ساختمان داده‌ها با	http://daneshnegar.com/book_۳۷۹۰۹۴.html	http://ktbr.ir/b۲۹۶۷۶	http://ketabesabz.com/dl/۵۲۱۵۶
طراحی سیستم‌های شی‌گرا با C# زبان	http://daneshnegar.com/book_۳۷۴۶۵۸.html	http://ktbr.ir/b۲۹۶۲۱	http://ketabesabz.com/dl/۵۲۱۲۶
مدیریت استراتژیک فناوری اطلاعات	http://daneshnegar.com/book_۳۷۴۶۵۹.html	http://ktbr.ir/b۲۹۷۷۹	http://ketabesabz.com/dl/۵۲۱۲۵
گرافیک رایانه‌ای با زبان C# برنامه‌نویسی	http://daneshnegar.com/book_۳۷۶۰۲۱.html	http://ktbr.ir/b۲۹۶۷۴	http://ketabesabz.com/dl/۵۱۰۴۹
درس و کنکور پایگاه داده پیشرفته	http://daneshnegar.com/book_۳۹۲۵۷۸.html	http://ktbr.ir/b۲۹۶۴۴	http://ketabesabz.com/dl/۵۲۱۰۲
فیزیک الکتریسیته	http://daneshnegar.com/book_۳۷۹۱۶۱.html	http://ktbr.ir/b۲۹۶۸۰	
تجارت الکترونیکی	http://daneshnegar.com/book_۳۷۹۱۸۸.html	http://ktbr.ir/b۲۹۶۱۹	
راهنمای کاربردی کاربری برای شبکه‌های OPNET شبیه‌سازی کامپیوتر	http://daneshnegar.com/book_۳۹۲۵۸۳.html	http://ktbr.ir/b۲۸۵۰۴	
درس و کنکور سیستم عامل پیشرفته	http://daneshnegar.com/book_۳۹۲۵۸۰.html	http://ktbr.ir/b۲۸۵۰۵	http://ketabesabz.com/dl/۵۲۱۸۰

شبکه‌های کامپیوتری با رویکرد کاربردی، آزمایشگاه شبیه‌سازی شبکه	http://daneshnegar.com/book_۳۹۲۲۵٤.html	http://ktbr.ir/b۲۸۵۲۸	
آزمایشگاه پایگاه دادهSQL با Server ۲۰۱۲	http://daneshnegar.com/book_۳۷۷۳۰۱.html	http://ktbr.ir/b۲۸۵۰۳	
کاربرد رایانه در مدیریت و حسابداری	http://daneshnegar.com/book_۳۷۵۸۹۲.html	http://ktbr.ir/b۲۸٤٥۰	
آموزش گام‌به‌گام برنامه‌نویسی بانک اطلاعاتی با ویژوال بیسیک‌نت	http://daneshnegar.com/book_۳٦۸۹۲۹.html	http://ktbr.ir/b۲۸٤٤۹	
آموزش گام‌به‌گام برنامه‌نویسی به زبان C++	http://daneshnegar.com/book_۳۸۰۲۳۸.html	http://ktbr.ir/b۲۸٤٥۲	
دانلود کتاب آموزش گام‌به‌گام برنامه‌نویسی بانک اطلاعاتی با C#	http://daneshnegar.com/book_۳٦۸٤۸٦.html	http://ktbr.ir/b۲۸٤٤۸	
حل مسائل پاسکال		http://ktbr.ir/b۲۸۳۹۸	
حل مسائل C++	http://daneshnegar.com/book_۳۹۲۲٦۲.html	http://ktbr.ir/b۲۸٤۰۱	http://ketabesabz.com/dl/٥۱۰٤۸
دانلود کتاب حل مسائل C#	http://daneshnegar.com/book_۳۷٤٦٥۷.html	http://ktbr.ir/b۲۸۳۹۹	http://ketabesabz.com/dl/٥۱۰۱۱
دانلود کتاب حل مسائل C		http://ktbr.ir/b۲۸۳۹۷	

فصل ۱ آشنایی با زبان C#

سازمان‌ها برای نگهداری داده‌ها از بانک اطلاعاتی استفاده می‌کنند. یکی از پرکاربردترین نرم‌افزارهای مدیریت بانک‌های اطلاعات، SQL Server است. از طرف دیگر، بانک اطلاعات به تنهایی نمی‌تواند نیازهای سازمان‌ها را برطرف کند. به همین دلیل با یکی از زبان‌های برنامه‌سازی باید بتوان به بانک اطلاعات متصل شده داده‌های آن را ویرایش، حذف و اضافه نمود. امروزه صدها زبان برنامه‌سازی وجود دارند که از طریق آنها می‌توان به بانک اطلاعات متصل گردید و داده‌های آن را دستکاری نمود. یکی از مهم‌ترین و پرکاربردترین زبان‌های برنامه‌سازی، C# می‌باشد. به همین دلیل در این فصل به طور خلاصه زبان C# را می‌آموزیم.

۱-۱. زبان C#

این زبان به همراه نرم‌افزار ویژوال استودیونت ارائه شده است. زبان C# از فناوری شیء[۱] و مفهوم شی‌گرایی[۲] استفاده می‌کند. هر چیزی که در دنیای واقعی وجود دارد، **شیء** نامیده می‌شود. مثل مردم، ساختمان‌ها، کارخانه‌ها، گیاهان، اتومبیل‌ها، کامپیوترها و غیره. هر شیء از **صفاتی** مانند اندازه، رنگ، وزن و دیگر صفات تشکیل شده است که شکل ظاهری آن را تعیین می‌کنند و رفتارهایی از خودشان نشان می‌دهند، مانند انسان می‌خوابد، گریه می‌کند، می‌خندد، اتومبیل حرکت می‌کند، ترمز می‌نماید و غیره.

از آنجایی که زبان C# از فناوری شی‌گرایی استفاده می‌نماید، امکاناتی در این زبان اضافه شده است که می‌توانید اشیای دنیای واقعی را مدل‌سازی کنید. برای این که C# شی‌گرایی را پیاده‌سازی کند از مفهوم **کلاس**[۳] استفاده می‌کند. **کلاس**، برای ایجاد انواع جدید به کار می‌رود. هر کلاس شامل داده‌ها و متدهایی است که داده‌ها را دستکاری می‌کنند و سرویس‌هایی را برای مشتریان فراهم می‌نمایند. با مفهوم کلاس و چگونگی پیاده‌سازی آنها در ادامه بیشتر آشنا خواهیم شد.

۱- Object ۲- Object Oriented ۳- Class

۲-۱. فضای نام

همان‌طور که بیان گردید، زبان برنامه‌نویسی C# از کلاس برای برنامه‌نویسی استفاده می‌کند. کلاس‌ها بـه دو دسته تقسیم می‌شوند که عبارتند از:

۱. کلاس‌های آماده: این کلاس‌ها از قبل نوشته شده‌اند و در کتابخانـه FCL ویـژوال اسـتودیونت وجـود دارند.

۲. کلاس‌هایی که برنامه‌نویس می‌نویسد: همه کلاس‌های مورد نیاز برنامه‌نویسان از قبل وجود ندارنـد. بنابراین، برنامه‌نویس نیاز دارد برخی از کلاس‌ها را بنویسد. در ادامه با این کلاس‌ها آشنا خواهید شد.

کلاس‌هایی که در کتابخانه FCL وجود دارند، در **فضاهای نام**[۱] مختلف قرار می‌گیرند. برخی از فضای نام‌ها و وظایف آنها در جدول ۱-۱ آمده است. این فضاهای نام به طور خودکار به برنامه C# اضافه می‌شوند. عـلاوه بر این فضاهای نام، فضاهای نام دیگری وجود دارند که می‌توانید آنها را به برنامه اضافه کنید. برای ایـن منظـور باید از دستور using استفاده نمایید. به عنوان مثال، دستورات زیر را ببینید:

```
using System.Data.SqlClient;
using System.Convert;
```

دستور اول، فضای نام System.Data.SqlClient را به برنامه اضافه می‌کند تا بتوانید از کلاس‌هایی که بـرای اتصال و دستکاری به بانک اطلاعات SQL Server به کارمی‌روند، بهره بگیرید (با این فضای نام در ادامه بیشتر آشنا خواهید شد) و دستور دوم، فضای نام System.Convert را به برنامه اضافه می‌کند تا بتوانید از متدهایی کـه برای تبدیل انواع داده‌های مختلف به کار می‌روند، استفاده نمایید.

جدول۱-۱ برخی از فضاهای نام موجود در FCL.

فضای نام	هدف
System	حاوی کلاس‌های پایه و انواع داده از قبیل double، int، char و غیره است.
System.Data	دارای کلاس‌هایی است که برای دستیابی به داده‌های بانک اطلاعات استفاده می‌شوند.
System.Io	از کلاس‌هایی تشکیل شده است که برای ورودی- خروجی داده‌ها مانند فایل‌هـا بـه کار می‌روند.
System.Drawing	از کلاس‌هایی تشکیل شده است که برای ترسیم اشکال گرافیکی به کار می‌روند.

۱- **Namespaces**

System.Linq	از کلاس‌هایی تشکیل شده است که برای کار با LINQ به کار می‌روند.

۳-۱. انواع داده‌ها

اکثر برنامه‌ها با دریافت داده‌ها، پردازش و استخراج نتایج سر و کار دارند. یعنی، داده‌ها مهم‌ترین بخش برنامه‌نویسی را ایفا می‌نمایند. لذا، باید انواع داده‌هایی که در زبان برنامه‌نویسی وجود دارند، را بی‌آموزیم. دو نوع داده را می‌توان در زبان C# تعریف نمود که عبارتند از:

۱. داده‌های مقدار **۲.** داده‌های مرجع

داده‌های مقدار، همان داده‌هایی هستند که توسط انواع اولیه تعریف می‌شوند (بجز داده‌های Object و String). این انواع در جدول ۲-۱ آمده‌اند و **داده‌های مرجع**، تمام انواعی هستند که توسط برنامه‌نویس تعریف می‌شوند و یا کلاس‌هایی هستند که از قبل تعریف شده‌اند. در ادامه پیاده‌سازی کلاس را می‌آموزیم.

جدول۲-۱ انواع داده‌های اولیه در C#.

نوع در زبان C#	معادل.NET	اندازه	نگهداری می‌کند.
byte	Byte	۱	مقادیر بدون علامت (۰ تا ۲۵۵)
char	Char	۱	کارکترهای یونیکد
bool	Boolean	۱	مقادیر True یا False
sbyte	Sbyte	۱	مقادیر علامت‌دار (۱۲۸- تا ۱۲۷)
short	Int۱٦	۲	مقادیر صحیح از ۳۲۷٦۸- تا ۳۲۷٦۷
ushort	Uint۱٦	۲	مقادیر صحیح بدون علامت از ۰ تا ٦۵۵۳۵
int	Int۳۲	٤	مقادیر صحیح بین $۲^{۳۱}$- تا ($۲^{۳۱}$ - ۱)
uint	Uint۳۲	٤	مقادیر صحیح بدون علامت ۰ تا $۲^{۳۲}$
float	Single	٤	اعداد اعشاری $\pm ۱/۵\times ۱۰^{-۴۵}$ تا $\pm ۳/۴\times ۱۰^{۳۸}$ با ۷ رقم اعشار
double	Double	۸	اعداد اعشاری $\pm ۵/۰\times ۱۰^{-۳۲۴}$ تا $\pm ۱/۷\times ۱۰^{۳۰۸}$ با ۱۵ تا ۱٦ رقم بعد از اعشار
decimal	Decimal	۸	نقطه اعشار ثابت تا ۲۸ رقم

long	Int٦٤	٨	مقادیر صحیح $-۲^{۶۳}$ تا $(۲^{۶۳}-۱)$
ulong	Uint٦٤	٨	مقادیر ۰ تا $(۲^{۶۴}-۱)$

۴-۱. متغیرها

متغیرها نامی برای کلمات حافظه هستند که دارای ویژگی‌های زیر می‌باشند:

- داده‌ها در آنها ذخیره می‌شوند.
- مقدار آنها در طول اجرای برنامه ممکن است تغییر کند.
- در یک لحظه خاص فقط یک مقدار دارند.

برای استفاده از متغیرها باید نام، نوع و مقدار آنها را تعیین کرد.

۱-۴-۱. نامگذاری متغیرها

برای نامگذاری متغیرها در C# می‌توان از ترکیبی از حروف، ارقام و خط ربط (_) استفاده کرد. به طوری که اولین کارکتر آنها رقم نباشد. به عنوان مثال، sum، count۱، i_۱ می‌توانند نام‌هایی برای متغیرها باشند، ولی ۱count و hioh!book نمی‌توانند نام متغیرها باشند. زیرا، اولین نام با رقم ۱ شروع شد و در دومین نام از کارکتر ! استفاده گردید که کارکتر مجاز نمی‌باشد.

۲-۴-۱. اعلان متغیرها

بعد از این که متغیرها را نامگذاری کردید باید نوع آنها را تعیین نمایید. یعنی، باید تعیین کنید متغیر چه نوع داده‌ای را ذخیره می‌کند. متغیرها به صورت زیر اعلان می‌گردند:

سطح دستیابی نوع داده نام متغیر ;

سطح دستیابی، تعیین می‌کند که در چه محدوده‌ای می‌توانید به متغیرها دستیابی داشته باشید و مهم‌ترین آنها public و private هستند. public موجب می‌شود تا بتوانید متغیر در کل کلاس دستیابی داشته باشید و private موجب می‌شود تا بتوان به متغیر در همان بلاک دستیابی داشته باشید. اگر سطح دستیابی ذکر نشود، private در نظر گرفته خواهد شد. در ادامه بیشتر با سطوح دستیابی آشنا خواهید شد.

اکنون دستورات زیر را ببینید:

```
int x;
double d;
string s1, s2;
```

دستور اول، متغیر x را از نوع صحیح تعریف می‌کند. دستور دوم، متغیر d را از نوع double و دستور سوم، متغیرهای s۱ و s۲ را از نوع رشته‌ای تعریف می‌نماید.

۳-۴-۱. مقدار دادن به متغیرها

برای مقداردهی به متغیرها روش‌های متعددی وجود دارد که عبارتند از:

۱. مقداردهی در زمان اعلان متغیر **۲.** پس از تعریف نوع متغیر و با دستور انتساب (=)

۳. دستورات ورودی

به عنوان مثال، دستورات زیر را ببینید:

```
int x = 5 , y = 10;
string s = "good";
```

دستور اول، متغیرهای x و y را با مقادیر ۵ و ۱۰ از نوع int تعریف می‌کند و دستور دوم، متغیر رشته‌ای s را از نوع string تعریف می‌نماید و رشته "good" را به آن تخصیص می‌دهد. اکنون دستورات زیر را ببینید:

```
int x;
string s;
x = 10;
s = "C#.NET";
```

دستورات اول و دوم متغیرهای x و s را به ترتیب از نوع int و string تعریف می‌کنند. دستور سوم، مقدار متغیر x را برابر ۱۰ قرار می‌دهد و دستور چهارم، مقدار متغیر s را رشته C#.NET تعیین می‌کند.

در ادامه با چگونگی مقداردهی متغیرها با دستورات ورودی آشنا خواهید شد.

۵-۱. ثوابت

ثوابت، مقادیری هستند که در برنامه وجود دارند ولی، قابل تغییر نیستند. برای تعریف ثوابت از واژه const به صورت زیر استفاده می‌شود:

const نوع داده نام = مقدار ثابت ;

به عنوان مثال، دستورات زیر را ببینید:

```
const float PI = 3.14;
const char ch = '+';
```

دستور اول، ثابت PI را با مقدار ۳,۱٤ از نوع اعشاری معرفی می‌کند و دستور دوم، ثابت ch را از نوع کارکتری با مقدار '+' تعریف می‌نماید.

۶-۱. عملگرها

عملگرها، نمادهایی هستند که اعمال خاصی را انجام می‌دهند. به عنوان مثال، نماد '-' عملگری است که دو مقدار را از یکدیگر تفریق می‌کند. عملگرها در C# به انواع مختلف تقسیم می‌شوند که در زیر آمده‌اند:

- **عملگرهای محاسباتی**، عملگرهایی هستند که عمل محاسبات را بر روی عملوندها انجام می‌دهند. این عملگرها در جدول ۳-۱ آمده‌اند.
- **عملگرهای رابطه‌ای**، عملگرهایی هستند که دو عملوند را با یکدیگر مقایسه می‌کنند (جدول ٤-۱ را ببینید).
- **عملگرهای منطقی**، عملگرهایی هستند که بر روی عبارات منطقی (True یا False) عمل می‌کنند (جدول ۵-۱ را مشاهده کنید).
- **عملگرهای ترکیبی**، از ترکیب عملگرهای محاسباتی و علامت = ایجاد می‌شوند (جدول٦-۱).
- **عملگرهای بیتی**، عملگرهایی هستند که برای تست کردن، مقداردهی یا شیفت دادن و سایر اعمال بر روی عملوندها به کار می‌روند. عملگرهای بیتی در جدول ۷-۱ آمده‌اند.

جدول ۳-۱ عملگرهای محاسباتی.

عملگر	نام	x	y	مثال	نتیجه
-	تفریق و منهای یکانی	۱۰	۱۲	y-x -y	۲ -۱۲
+	جمع	۱۷	۵	x +y	۲۲
*	ضرب	۱۰	۲	x*y	۲۰
/	تقسیم	۱۰	۵	x/y	۲
%	باقیمانده تقسیم صحیح	۱۰	۳	x%y	۱
--	کاهش	۱۰	۵	--x , y--	٤ , ۹
++	افزایش	۱۰	۵	x++ , ++y	٦ , ۱۱

جدول ۷-۱ عملگرهای بیتی.

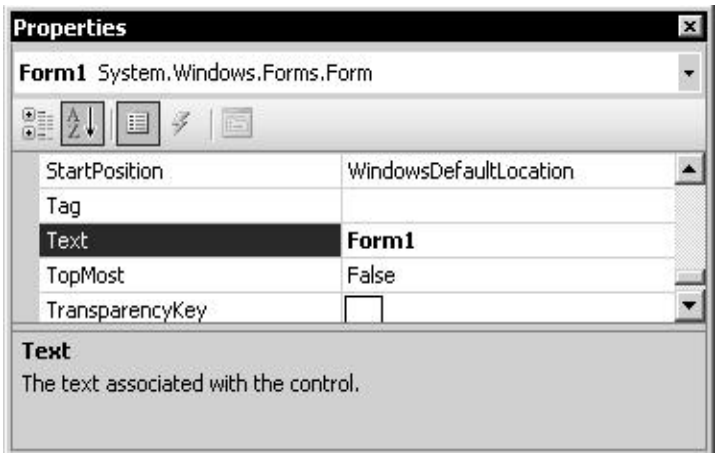
عملگر	نام	x	y	مثال	نتیجه
&	و	۱۰	۲	x & y	۲
|	یا	۱۰	۳	x | y	۱۱
^	یا انحصاری	۱۰	۳	x ^ y	۹
~	نقیض	۱۲۶	۳	~ x	۱
>>	شیفت به راست	۵۶	۳	x >> y	۷
<<	شیفت به چپ	۲۵	۲	x << y	۱۰۰

۷-۱. فرم برنامه

فرم برنامه، مکانی است که کنترل‌های برنامه در آن قرار می‌گیرند. هر برنامه در C# حداقل یک فرم دارد. برای استفاده از فرم باید خواص، رویدادها و متدهای آن را بشناسیم. لذا، در ادامه به این موضوعات می‌پردازیم.

۱-۷-۱. خواص فرم

خواص فرم، شکل ظاهری فرم را تعیین می‌کنند. فرم خواص متعددی دارد. بیان همه این خواص از حوصله این کتاب خارج است. لذا، به تشریح خواص مهم فرم و کنترل‌های دیگر می‌پردازیم. برخی از خواص مهم فرم در جدول ۸-۱ آمده است. برای نمایش خواص فرم، بر روی آن کلیک کنید و سپس کلید F4 را بزنید تا خواص فرم را ببینید (شکل زیر):



جدول ۸-۱ خواص مهم فرم.

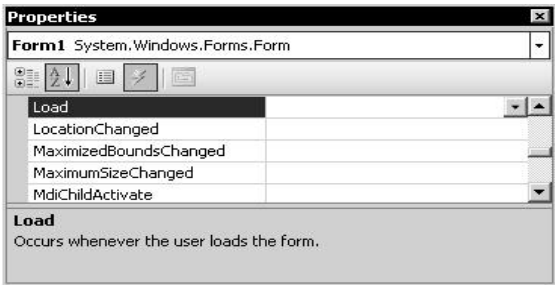
خاصیت	هدف
Name	نام فرم را تعیین می‌کند. برای اولین فرم، نام فرم Form۱ است که می‌توانید آن را تغییر دهید.
ActiveForm	فرم فعال فعلی را تعیین می‌کند.
Enabled	تعیین‌می‌کند آیا فرم فعال است یا خیر. اگر فرم غیرفعال گردد، به‌هیچ‌رویدادی‌پاسخ نمی‌دهد.
Font	فونت فرم را تعیین می‌کند.
ForeColor	رنگ‌نوشته‌های‌روی‌فرم را تعیین می‌کند.
Language	زبان مورد استفاده در فرم را تعیین می‌کند.
MinimizeBox	تعیین می‌کند آیا دکمه کمینه در عنوان فرم ظاهر شود یا خیر.
RightToLeft	جهت‌نمایش‌اطلاعات‌را تعیین‌می‌کند(این خاصیت برای زبان فارسی مفید است).
Size	اندازه فرم را تعیین می‌کند.
Text	متنی را تعیین می‌کند که باید در عنوان فرم نمایش داده شود.
Location	محل قرار گرفتن فرم را تعیین می‌کند.

۲-۷-۱. رویدادهای فرم

همان‌طور که بیان کردیم، برنامه‌های ویژوال منتظر رخ دادن رویدادهایی هستند که توسط کاربر انتخاب می‌شوند تا به آنها پاسخ دهند. یعنی، اگر برنامه پاسخ‌گو به رویدادی نوشته شده باشد، در صورت انتخاب آن

رویداد توسط کاربر، این برنامه اجرا می‌شود. فرم دارای رویدادهای مختلفی است. برخی از مهم‌ترین آنها در جدول ۹-۱ آمده‌اند.

برای مشاهده رویدادهای فرم، بر روی آن کلیک کنید تا فرم انتخاب شود. اکنون گزینه View/ Properties Window را اجرا نمایید تا پنجره Properties ظاهر شود. در این پنجره دکمه Events را کلیک کنید تا لیست رویدادهای فرم را ببینید (شکل زیر):



جدول ۹-۱ رویدادهای مهم فرم.

رویداد	هدف
Activated	وقتی که فرم فعال می‌شود، رخ می‌دهد.
Click	وقتی رخ می‌دهد که فرم کلیک گردد.
FormClosed	وقتی رخ می‌دهد که فرم بسته شود.
FormClosing	وقتی رخ می‌دهد که فرم در حال بسته شدن باشد. این رویداد قبل از رویداد FormClosed رخ می‌دهد.
Deactivate	وقتی رخ می‌دهد که فرم غیرفعال گردد.
DoubleClick	وقتی که فرم کلیک مضاعف شود، رخ می‌دهد.
Enter	وقتی مکان‌نما وارد فرم می‌شود، رخ می‌دهد.
KeyDown	وقتی کلیدی فشرده شده پایین می‌رود، رخ می‌دهد.
KeyPress	وقتی کلیدی فشرده می‌شود، رخ می‌دهد. این رویداد قبل از رویداد KeyDown اتفاق می‌افتد.
KeyUP	وقتی کلید فشرده شده رها می‌گردد، رخ می‌دهد.
Leave	وقتی مکان‌نما فرم را ترک می‌کند رخ می‌دهد.

Load	وقتی فرمی باز می‌شود، رخ می‌دهد (قبل از نمایش فرم).
MouseDown	وقتی کلید ماوس فشرده شود، رخ می‌دهد.
MouseEnter	وقتی مکان‌نمای ماوس وارد فرم شود، رخ می‌دهد.
MouseLeave	وقتی مکان‌نمای ماوس فرم را ترک می‌کند، رخ می‌دهد.
MouseUP	وقتی کلید فشرده شده ماوس رها می‌شود، رخ می‌دهد.
Move	وقتی فرم شروع به حرکت می‌کند، رخ می‌دهد.
Resize	وقتی اندازه فرم تغییر می‌یابد، رخ می‌دهد.
TextChanged	وقتی رخ می‌دهد که متن فرم (کنترل) یا خاصیت Text تغییر کند.
Shown	وقتی که فرم نمایش داده شود، رخ می‌دهد.
SizeChanged	وقتی رخ می‌دهد که مقدار خاصیت Size تغییر می‌یابد.
MouseMove	وقتی رخ می‌دهد که ماوس روی فرم حرکت می‌کند.

۳-۷-۱. متدهای فرم

متدها، کارهای خاصی را بر روی فرم انجام می‌دهند. برخی از متدهای مهم فرم در جدول ۱۰-۱ آمده‌اند.

جدول ۱۰-۱ متدهای مهم فرم.

متد	هدف
Active	فرم را فعال می‌کند.
Close	فرم را می‌بندد.
Dispose	فرم را حذف کرده از بین می‌برد.
Focus	مکان‌نما را به فرم مورد نظر منتقل می‌کند.
Hide	فرم را پنهان می‌کند.
ResetText	متن عنوان فرم را پاک می‌کند.
Show	فرم مخفی شده را آشکار می‌کند.
Refresh	فرم را بازسازی کرده، اطلاعات آن را دوباره رسم می‌کند.
ToString	محتویات فرم را به رشته تبدیل می‌نماید.

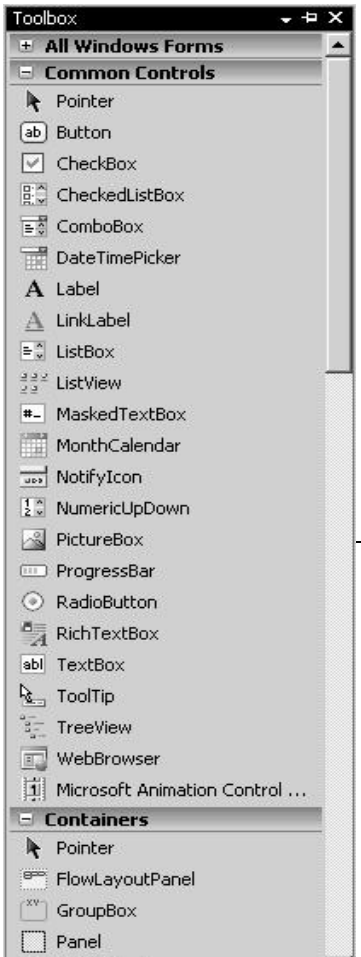
Validate	موجب اعتبارسنجی فرم می‌شود.

۸-۱. کنترل‌ها

همان‌طور که می‌دانید برای نوشتن برنامه‌ها در C# باید کنترل‌هایی را بر روی فرم قرار دهید (این کنترل‌ها همان قطعات تشکیل دهنده برنامه هستند). خواص آنها را مقداردهی کرده، برنامه پاسخ‌گو به رویدادهای آنها را بنویسید. بنابراین، کنترل‌ها اجزا اصلی برنامه‌های C# را تشکیل می‌دهند. کنترل‌های زیادی در C# وجود دارند. بحث در مورد همه اینها در این کتاب نمی‌گنجد. لذا، در این کتاب برخی از کنترل‌های مهم را بررسی می‌کنیم.[۵]

این کنترل‌ها را در ادامه خواهید دید. برای اضافه کردن کنترل جدید بر روی فرم، دکمه Toolbox () را کلیک کنید تا کنترل‌های آماده را مشاهده کنید. اکنون جعبه ابزار ظاهر می‌شود (شکل ۱-۱). در این جعبه ابزار، کنترل مورد نظر را کلیک مضاعف کنید تا به فرم اضافه گردد و آن را به مکان دلخواه از فرم انتقال دهید (با کشیدن و رها کردن). من دکمه button۱ را به فرم اضافه کردم و به مکان دلخواه انتقال دادم (شکل زیر):

برای حذف کنترلی از فرم، آن را کلیک کرده تا انتخاب شود. اکنون دکمه Delete را فشار دهید تا کنترل از روی فرم حذف گردد.



۱-۸-۱. کنترل Label ()

این کنترل برای نمایش متن‌های غیرقابل ویرایش از قبیل عنوان فیلد، پیام‌های مورد نیاز به کار می‌رود. این کنترل نیز دارای خواص و رویدادهای زیادی است که برخی از مهم‌ترین آنها در زیر آمده است:

- **خاصیت AutoSize:** تعیین می‌کند آیا اندازه کنترل با توجه به مقدار خاصیت Text به طور خودکار تغییر کند یا خیر.

۱- برای کسب اطلاعات بیشتر در مورد C# به آموزش گام به گام C# از انتشارات علوم رایانه، مراجعه کنید.

- **خاصیت TextAlign**: مکان قرار گرفتن متن در کنترل Label را تعیین می‌کند. به عنوان مثال، دستور زیر را ببینید.

```
label1.TextAlign = ContextAlignment.TopLeft;
```

این دستور، متن را در بالا سمت چپ نمایش می‌دهد.

- **رویداد AutoSizeChanged**: وقتی رخ می‌دهد که مقدار خاصیت AutoSize تغییر کند.
- **رویداد TextAlignChanged**: وقتی رخ می‌دهد که مقدار خاصیت TextAlign تغییر یابد.

۲-۸-۱. کنترل TextBox (TextBox)

این کنترل برای دریافت اطلاعاتی از قبیل مقادیر رشته‌ای، عددی و Memo (اطلاعات رشته‌ای طولانی) فیلدها به کار می‌رود. این کنترل نیز مانند کنترل‌های دیگر دارای خواص، رویدادها و متدهای متعددی است. برخی از خواص، رویدادها و متدهای این کنترل در جدول ۱-۱۱ آمده‌اند.

۳-۸-۱. کنترل Button (Button)

این کنترل‌ها به دکمه فرمان معروف هستند. زیرا با کلیک آن فرمان خاصی (دستورات خاصی) اجرا خواهد شد. خواص، رویدادها و متدهای این کنترل مانند کنترل‌های دیگر است. در ادامه بیشتر با این کنترل آشنا خواهید شد.

۴-۸-۱. کنترل ListBox (ListBox)

این کنترل برای نمایش لیستی از اشیاء به کار می‌رود. در این کنترل می‌توان اطلاعاتی از قبیل نام افراد، لیست کالاها و غیره را نمایش داد. این کنترل نیز مانند کنترل‌های دیگر دارای خواص، رویدادها و متدهای زیادی است که برخی از پرکاربردترین آنها در جدول ۱-۱۲ آمده‌اند.

جدول ۱-۱۱ خواص، رویدادها و متدهای کنترل TextBox.

خاصیت	هدف
Lines	رشته‌ای از نوع آرایه است که تعداد خط‌های TextBox را تعیین می‌کند.
MaxLength	حداکثر طول متن را تعیین می‌کند. به عنوان مثال، اگر در این خاصیت ۵۰ را وارد کنید، کاربر حداکثر می‌تواند ۵۰ کاراکتر را وارد کند.
MultiLines	تعیین می‌کند آیا TextBox می‌تواند چند سطر اطلاعات را بپذیرد یا خیر.
Readonly	تعیین می‌کند که آیا محتویات TextBox فقط خواندنی باشد یا خیر. اگر اطلاعات TextBox فقط خواندنی باشد، این کنترل مانند Label عمل می‌کند.
PasswordChar	تعیین می‌کند در هنگام ورود اطلاعات در TextBox آیا اطلاعات به صورت کلمه عبور ظاهر شوند (اطلاعات اصلی مخفی گردند یا خیر).
رویداد	**هدف**
MultiLineChanged	زمانی رخ می‌دهد که خاصیت MultiLines تغییر یابد.
ReadonlyChanged	زمانی رخ می‌دهد که خاصیت Readonly تغییر یابد.
متد	**هدف**
Clear	محتویات TextBox را پاک می‌کند.
ClearUndo	اثر اجرای فرمان Clear انجام شده را خنثی می‌کند.
Copy	متن انتخاب شده در TextBox را در حافظه موقت کپی می‌کند.
Cut	متن انتخاب شده در TextBox را به حافظه موقت انتقال می‌دهد.
Paste	متن حافظه موقت را در مکان فعلی کپی می‌نماید.
Undo	تأثیر آخرین فرمان انجام شده را خنثی می‌کند.
Select	برای انتخاب متن TextBox به کار می‌رود.
SelectAll	کل متن TextBox را انتخاب می‌کند.
DeselectAll	کلیه متن انتخاب شده TextBox را از حالت انتخاب خارج می‌کند.

اکنون دستورات زیر را ببینید:

```
listBox1.Clear();
listBox1.Sorted = true;
```

```
listBox1.Items.Add ("one");
listBox1.Items.Insert ("zero" , 0);
listBox1.Items.RemoveAt (1);
```

دستور اول، گزینه‌های listBox۱ را حذف می‌کند. دستور دوم، گزینه‌های listBox۱ را به صورت مرتب شده نمایش می‌دهد. دستور سوم، گزینه one را به listBox۱ اضافه می‌کند. دستور چهارم، گزینه "zero" را قبل از گزینه "one" اضافه می‌نماید و دستور پنجم، مقدار مکان دوم listBox۱ را حذف می‌کند.

۵-۸-۱. کنترل ComboBox

این کنترل ترکیبی از یک کنترل TextBox و ListBox است که برای تایپ یا انتخاب گزینه‌ای به کار می‌رود. خواص، رویدادها و متدهای این کنترل مانند کنترل ListBox است.

۶-۸-۱. کنترل CheckBox (CheckBox)

این کنترل برای تعریف گزینه‌هایی با دو انتخاب از قبیل مرد یا زن ، جانباز یا غیرجانباز ، متأهل یا مجرد به کار می‌رود. این کنترل را می‌توان به فیلدهای منطقی در بانک اطلاعات انقیاد[۱] کرد. برخی از خواص و رویدادهای کنترل CheckBox در زیر آمده‌اند:

- **خاصیت CheckAlign**: محل قرار گرفتن ✓ (تیک) را در کنار مربع تعیین می‌کند.
- **خاصیت Checked**: تعیین می‌کند آیا کنترل CheckBox انتخاب شده است یا خیر (✓ نمایش داده شده است یا خیر).
- **خاصیت CheckState**: یکی از سه حالت CheckBox را تعیین می‌کند که می‌تواند مقادیر Unchecked (انتخاب نشده)، Checked (انتخاب شده) یا Indeterminate (غیرفعال) را بپذیرد.
- **خاصیت TreeState**: تعیین می‌کند آیا کنترل CheckBox دارای سه حالت است یا خیر.
- **رویداد CheckedChanged**: وقتی رخ می‌دهد که خاصیت Checked تغییر یابد.
- **رویداد CheckStateChanged**: وقتی رخ می‌دهد که خاصیت CheckState تغییر یابد.

۱- **Bind**

۷-۸-۱. کنترل CheckedListBox (CheckedListBox)

این کنترل ترکیبی از CheckBox و ListBox است. یعنی، هر یک از گزینه‌های ListBox دارای حالت انتخاب CheckBox می‌باشند. خواص و رویدادهای این کنترل مانند رویدادهای CheckBox و ListBox است. ولی، دو متد زیر به این کنترل اضافه شده است:

- **متد SetItemsChecked**: برای مقداردهی به خاصیت Checked این کنترل به کار می‌رود. به عنوان مثال، دستور زیر را ببینید.

```
CheckedListBox1.SetItemsChecked (3, true);
```

این دستورحالت چهارمین گزینه انتخاب شده CheckListBox۱ را تیک‌دار می‌کند (اندیس گزینه‌ها از صفر شروع می‌شود).

- **متد SetItemsCheckState**: مقدار خاصیت CheckState گزینه خاصی را تعیین می‌کند.

۸-۸-۱. کنترل RadioButton (RadioButton)

این کنترل، دکمه‌های رادیویی را ایجاد می‌کند که می‌توان فقط یک گزینه (یکی از آنها) را انتخاب کرد. یعنی، برای ایجاد دکمه‌هایی به کار می‌رود که دارای گزینه‌های ناسازگار هستند. چنانچه بخواهید چند گروه از RadioButton داشته باشید، آنها را باید در کنترل‌های GroupBox مختلف اضافه کنید.

۹-۸-۱. کنترل GroupBox (GroupBox)

این کنترل برای ایجاد گروه‌های متعدد کنترل‌ها به کار می‌رود. اگر بخواهید چند کنترل را در یک گروه قرار دهید، باید آنها را به یک کنترل GroupBox اضافه کنید. خواص، متدها و رویدادهای این کنترل مانند کنترل‌های دیگر است.

۱۰-۸-۱. کنترل MenuStrip (MenuStrip)

این کنترل برای ایجاد منو به کار می‌رود. چنانچه این کنترل را به فرم اضافه کنید. شکل ۲-۱ ظاهر می‌شود که می‌توانید گزینه‌های منو را در بخش Type Here تایپ کنید. در مثال ۱ - ۱، چگونگی اضافه کردن منو را خواهید دید. این کنترل دارای خواص زیر است:

- **خاصیت Enabled**: تعیین می‌کند آیا گزینه منو فعال باشد یا غیرفعال.

- **خاصیت Checked**: تعیین می‌کند آیا گزینه منو می‌تواند دارای علامت تیک (✓) باشد یا خیر.
- **خاصیت Shortcut**: کلید میانبری را برای گزینه منو تعیین می‌کند.
- **خاصیت Text**: عنوان گزینه منو را تعیین می‌کند، ولی اگر در این خاصیت مقدار - را وارد کنید، گزینه منو به عنوان یک جدا کننده در نظر گرفته می‌شود.

شکل ۲-۱ اضافه کردن منو.

۱۱-۸-۱. کنترل ContextMenuStrip (ContextMenuStrip)

این کنترل همانند کنترل MenuStrip می‌باشد، با این تفاوت که برای ایجاد منوی میانبر به کار می‌رود. منو میانبر، منویی است که با کلیک راست بر روی کنترل نمایش داده می‌شود و می‌توان گزینه‌های آن را اجرا نمود. خواص، رویدادها و متدهای این کنترل مانند کنترل MenuStrip است.

۱۲-۸-۱. کنترل PictureBox (PictureBox)

این کنترل برای نمایش تصاویری از قبیل نمادهای گرافیکی، تصاویر بیت نگاشت[۷]، شبه فایل، آیکن‌ها و غیره به کار می‌رود. برخی از خواص و متدهای این کنترل در زیر آمده‌اند:

- **خاصیت Image**: تصویری را تعیین می‌کند که باید در کنترل PictureBox نمایش داده شود. به عنوان مثال، دستورات زیر را ببینید:

```
Image img1 = new Bitmap ("D:\\1.gif");
PictureBox1.Image = (Image) img1;
```

این دستورات، فایل ۱.gif ریشه درایو D را در کنترل PictureBox۱ نمایش می‌دهند.

۱- **Bitmap**

- **خاصیت SizeMode**: نحوه و اندازه نمایش تصویر را در PictureBox تعیین می‌کند و می‌تواند مقادیر زیر را بپذیرد:

۱. **مقدار Normal**: اندازه تصاویر را تغییر نمی‌دهد (تصویر را در اندازه واقعی نمایش می‌دهد).

۲. **مقدار StretchImage**: تصویر را به اندازه PictureBox بزرگ یا کوچک می‌کند.

۳. **مقدار AutoSize**: کنترل PictureBox را به اندازه تصویر تغییر می‌دهد.

٤. **مقدار CenterImage**: تصویر را در وسط کنترل PictureBox نمایش می‌دهد.

- **متد Save**: برای ذخیره تصویر موجود در کنترل PictureBox بر روی حافظه جانبی به کار می‌رود. به عنوان مثال، دستور زیر را ببینید:

```
PictureBox1.Image.Save ("test.gif");
```

تصویر موجود در PictureBox۱ را در فایل test.gif ذخیره می‌کند.

۹-۱. ساختارهای کنترلی

در برنامه‌های ساده، دستورات برنامه به صورت پشت سر هم (از اولین دستور به آخرین دستور) اجرا می‌شوند. گاهی نیاز است بعضی از دستورات چندین بار اجرا شوند، تحت شرایط خاصی اجرا شده یا اجرا نگردند. برای پیاده‌سازی چنین برنامه‌هایی از ساختارهای کنترلی استفاده می‌شود. این ساختارها دو نوع‌اند که عبارتند از:

۱. ساختارهای تصمیم

۲. ساختارهای تکرار

۹-۱-۱. ساختارهای تصمیم

این ساختارها برای حالتی به کار می‌روند که بخواهید تحت شرایطی مجموعه‌ای از دستورات اجرا شوند یا برخی دیگر اجرا نگردند. ساختارهای تصمیم در C# عبارتند از: if و switch.

ساختار تصمیم if

این ساختار یک عبارت شرطی را ارزیابی می‌کند، در صورتی که نتیجه ارزیابی شرط دارای ارزش درست باشد، مجموعه‌ای از دستورات اجرا می‌شوند، وگرنه، مجموعه دیگری از دستورات اجرا خواهند شد. این ساختار به صورت زیر به کار می‌رود:

```
if   (عبارت شرطی)
   {
        مجموعه دستورات۱ ;
   }
else
   {
        مجموعه دستورات ۲ ;
   }
```

در این ساختار، ابتدا **عبارت شرطی داخل پرانتز** ارزیابی می‌گردد، اگر نتیجه ارزیابی درست باشد، **مجموعه دستورات۱** اجرا می‌شوند، وگرنه **مجموعه دستورات۲** اجرا خواهند شد.

در این ساختار به نکات زیر توجه کنید:

- در هر یک از بخش‌های if و else اگر تعداد دستورات یکی باشد، بلاک‌های } و { را می‌توانید حذف کنید.
- در این ساختار می‌توان بخش else را حذف کرد. در این صورت، اگر نتیجه **ارزیابی عبارت شرطی** نادرست باشد، دستورات بعد از { if اجرا خواهند شد.
- در این ساختار، عبارت شرطی می‌تواند با عملگرهای منطقی از قبیل && یا ||، شرط‌های مختلف را ترکیب کرد.

ساختار if تودرتو

گاهی ممکن است بخواهید شرط‌های متعددی را بررسی کنید. برای این منظور، می‌توانید از ساختار if تودرتو استفاده کنید. این ساختار به صورت زیر به کار می‌رود:

```
if       (عبارت شرطی۱)
 {
      مجموعه دستورات ۱ ;
```

```
}
else if     (عبارت شرطی۲)
 {
     مجموعه دستورات ۲ ;
 }
     ⋮
else if   (عبارت شرطیn)
 {
 مجموعه دستوراتn ;
 }
else
 {
     مجموعه دستورات۱+n ;
 }
```

در این ساختار، اگر نتیجه **عبارت شرطی۱**، درست باشد، **مجموعه دستورات۱** اجرا می‌شوند و کنترل اجرای برنامه به بعد از { مربوط به else انتقال می‌یابد. وگرنه، اگر **نتیجه عبارت شرطی۲**، درست باشد، **مجموعه دستورات۲** اجرا می‌شوند و دستور if خاتمه می‌یابد و این روند ادامه می‌یابد و اگر هیچ یک از عبارت‌های شرطی نتیجه درستی نداشته باشند، **مجموعه دستورات ۱+n**، اجرا خواهند شد.

ساختار switch

ساختار تصمیم تودرتو، موجب کاهش خوانایی برنامه خواهد شد. برای رفع این مشکل، می‌توان از ساختار switch استفاده نمود. این ساختار به صورت زیر به کار می‌رود:

```
switch  (عبارت) {
case <مقدار۱>  :
    دستورات ۱ ;
break ;
case  <مقدار۲>  :
    دستورات۲ ;
break ;
    ⋮
```

```
case <مقدارn> :
    دستورات n ;
break ;
default :
    دستورات n+۱ ;
break ;
}
```

در این ساختار، ابتدا عبارت داخل پرانتز switch، ارزیابی می‌شود. اگر نتیجه ارزیابی این عبارت، برابر با **مقدار۱** باشد، **دستورات۱** اجرا خواهند شد و دستور break ساختار switch را خاتمه می‌دهد. اگر نتیجه ارزیابی عبارت برابر **مقدار ۱** نباشد، با **مقدار۲** مقایسه می‌گردد، اگر برابر این مقدار باشد، **دستورات۲** اجرا می‌شوند و ساختار switch خاتمه می‌یابد و این روند ادامه می‌یابد. اگر نتیجه **ارزیابی عبارت** برابر هیچ یک از مقادیر ۱ تا n نباشد، دستورات n+۱ اجرا خواهند شد.

در ساختار switch باید به نکات زیر دقت کنید:

- مقادیر موجود در caseهای دستور switch نمی‌توانند با هم مساوی باشند.
- اگر در یک case دستور break ذکر نشود، این مقدار case با مقدار case بعدی or (||) می‌گردد.
- دستور switch فقط مساوی بودن را بررسی می‌کند. ولی، ساختار if هر یک از شرط‌های منطقی (عملگرهای منطقی) را بررسی می‌نماید.

۱۰-۱. ساختارهای تکرار

برای انجام کارهای تکراری در برنامه از ساختارهای تکرار استفاده می‌شود. در C# حلقه‌های تکرار متعددی وجود دارند که برخی از آنها عبارتند از: for، while، do while و foreach.

ساختار تکرار for

این ساختار برای حالتی به کار می‌رود که تعداد تکرار از قبل مشخص باشد. این ساختار به صورت زیر به کار می‌رود:

```
for (گام حرکت ; شرط حلقه ; مقدار اولیه اندیس حلقه) {
    مجموعه دستورات بدنه حلقه ;
}
```

در این ساختار، ابتدا مقدار اولیه در اندیس حلقه قرار می‌گیرد. سپس شرط حلقه تست می‌گردد، اگر شرط دارای ارزش درستی باشد، **مجموعه دستورات** بدنه حلقه اجرا می‌گردند و گام حرکت به **اندیس حلقه** اضافه می‌شود. در ادامه شرط حلقه تست می‌گردد و این روند ادامه می‌یابد. تا زمانی که شرط حلقه نقض نگردد، حلقه ادامه می‌یابد. به عنوان مثال، دستورات زیر را ببینید.

```
int sum = 0;
for (i = 1 ; i <= 100 ; i++)
{
   sum += i;
   listBox1.Items.Add (i.ToString());
}
textBox1.Text = sum.ToString();
```

این دستورات، اعداد ۱ تا ۱۰۰ را به listBox۱ اضافه کرده مجموع آنها را در textBox۱ نمایش می‌دهند.

ساختار تکرار while

این ساختار برای زمانی به کار می‌رود که تعداد تکرار از قبل مشخص نباشد و به صورت زیر استفاده می‌شود:

while (شرط)
{
مجموعه دستورات بدنه حلقه ;
}

در این ساختار، ابتدا شرط داخل پرانتز ارزیابی می‌شود. اگر **شرط درست** باشد، **مجموعه دستورات بدنه حلقه** اجرا می‌شوند و شرط مجدداً ارزیابی می‌گردد و این روند تا زمانی که شرط ارزش درستی داشته باشد، ادامه می‌یابد و به محض اینکه شرط نقض (نادرست) شود، حلقه خاتمه می‌یابد. چنانچه مجموعه دستورات بدنه حلقه، یک دستور باشد، می‌توانید } و { را حذف نمایید.

ساختار تکرار do while

این ساختار همانند ساختار while است. با این تفاوت که **دستورات بدنه حلقه حداقل** یک بار اجرا می‌گردند. چون، شرط در انتهای حلقه ارزیابی (تست) می‌شود. این ساختار به صورت زیر به کار می‌رود:

```
do
{
مجموعه دستورات بدنه حلقه ;
}
while (شرط) ;
```

دستور break

این دستور موجب خروج از حلقه تکرار یا switch می‌شود و به صورت زیر به کار می‌رود:

break ;

دستور continue

این دستور، کنترل اجرای برنامه را به ابتدای حلقه تکرار برمی‌گرداند و به صورت زیر به کار می‌رود:

continue;

به عنوان مثال، دستورات زیر را ببینید:

```
listBox1.Items.Clear();
for (int i = 1; i <= 10; i++)
{
   if (i < 5) continue;
   listBox1.Items.Add (i.ToString());
}
```

این دستورات، مقادیر **۵** تا **۱۰** را به listBox۱ اضافه می‌کنند. زیرا، اگر i کوچک‌تر از **۵** باشد، کنترل اجرای برنامه به ابتدای حلقه برمی‌گردد.

۱۱-۱. مدیریت صفحه کلید

کنترل‌ها علاوه بر این که به رویدادهای ماوس پاسخ می‌دهند، می‌توانند به رویدادهای صفحه کلید از قبیل KeyPress، KeyDown و KeyUp پاسخ دهند. رویداد KeyPress قبل از رویداد KeyDown و رویداد KeyDown قبل از رویداد KeyUp اتفاق می‌افتد. **رویداد KeyPress**، زمانی اتفاق می‌افتد که کاربر کلیدی (به جز کلیدهای Tab، مکان‌نما و کلیدهایی که کداسکی آنها بین ۰ تا ۳۱ است) را فشار دهد. این رویداد دو پارامتر دارد که عبارتند از:

- **Sender**: از نوع Objcet است و شیءای را تعیین می‌کند که این رویداد بر روی آن رخ داده است.

- **e**: از نوع KeyEventArgs است و دارای ساختاری می‌باشد که اطلاعات کلید فشرده شده از قبیل KeyChar (مقدار کارکتر فشرده شده) و Handled (گرداننده رویداد KeyPress) را نگهداری می‌کند.
- **رویداد KeyDown**: وقتی کاربر کلیدی را فشار می‌دهد، این رویداد اتفاق می‌افتد (این رویداد به کلیدهای مکان‌نما، Tab و کارکترهایی که کد آنها بین ۰ تا ۳۱ باشد، نیز پاسخ می‌دهد). این رویداد همچنین مانند رویداد KeyPress دارای دو پارامتر sender و e است. ساختار پارامتر e در جدول ۱۳-۱ آمده است.
- **رویداد KeyUp**: زمانی رخ می‌دهد که کلید فشرده شده را رها کنید. پارامترهای این رویداد مانند رویداد KeyDown است.

جدول ۱۳-۱ ساختار پارامتر e مربوط به رویدادهای KeyUp و KeyDown.

خاصیت	هدف
Alt	آیا کلید Alt فشرده شده است یا خیر؟
Handled	گرداننده رویداد KeyDown و KeyUp را تعیین می‌کند.
Ctrl	آیا کلید Ctrl فشرده شده است یا خیر؟
KeyCode	کداسکی کارکتر فشرده شده را مشخص می‌کند.
KeyData	داده مربوط به کارکتر فشرده شده را تعیین می‌کند.
KeyValue	کارکتر فشرده شده را مشخص می‌نماید.
Shift	آیا کلید Shift فشرده شده است یا خیر؟

۱۲-۱. آرایه‌ها

تاکنون متغیرهایی که در برنامه‌ها استفاده کردیم، یک مقدار را ذخیره می‌کردند. گاهی نیاز است چندین مقدار با یک نام و از یک نوع داده را به صورت پشت سر هم در حافظه ذخیره کنید. برای این منظور باید متغیرهای **اندیس‌دار** یا **آرایه**‌ها را تعریف نمایید. آرایه با توجه به تعداد اندیس‌هایشان به انواع مختلف تقسیم می‌شوند. آرایه‌ای با یک اندیس را **آرایه یک بعدی**، آرایه دارای دو اندیس را **آرایه دو بعدی** و آرایه‌ای که n اندیس داشته باشد، **آرایه n بعدی** نام دارد. برای استفاده از آرایه دو کار باید انجام شود:

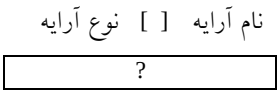
۱. اعلان آرایه

۲. تخصیص فضا به آرایه

ا علان آرایه یک بعدی به صورت زیر انجام می‌شود:

; نام آرایه [ ] نوع آرایه

این دستور، شکل زیر را ایجاد می‌کند:



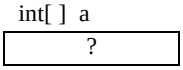
برای تخصیص فضا به آرایه باید نمونه‌هایی از آن را ایجاد کنید. این کار با دستور new و به صورت زیر انجام می‌شود:

; [تعداد عناصر آرایه] نوع آرایه new = نام آرایه

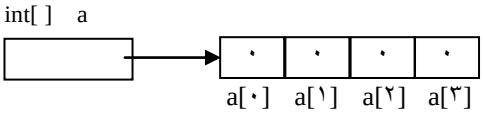
اکنون دستورات زیر را در نظر بگیرید:

```
int[] a;
a = new int[4];
```

دستور اول، آرایه a را از نوع int تعریف می‌کند (شکل زیر):



دستور دوم، آرایه‌ای با ٤ عنصر تعریف کرده، آدرس شروع آن را به a تخصیص می‌دهد (مانند شکل زیر):



در هنگام استفاده از آرایه به نکات زیر توجه داشته باشید:

۱. اندیس آرایه از صفر (۰) شروع می‌شود.

۲. تعداد عناصر آرایه را می‌توان در زمان اجرا تعیین کرد. یعنی، آرایه‌ای به صورت پویا تعریف نمود. به عنوان مثال، دستورات زیر را ببینید:

```
int count = Convert.ToInt32 (textBox1.Text);
int[] a = new int [count];
```

این دستورات، آرایه‌ای به نام a تعریف می‌کنند که تعداد عناصر آن از طریق کنترل textBox۱ تعیین می‌گردد.

۳. همان‌طور که می‌دانید، آرایه متغیری اندیس‌دار است. بنابراین، برای بازیابی و مقداردهی (دستیابی) به عناصر آن از اندیس به صورت زیر استفاده می‌شود:

[اندیس آرایه] نام آرایه

اکنون دستورات زیر را ببینید:

```
int[] a = new int[3];
a [0] = 10;
a [1] = 12;
a [2] = a [۰] + a [۱];
```

دستور اول، آرایه‌ای به نام a با ۳ عنصر تعریف می‌کند. دستور دوم، اولین عنصر آرایه (اندیس صفر) را برابر ۱۰ قرار می‌دهد. دستور سوم، دومین عنصر آرایه را برابر ۱۲ قرار داده و چهارمین دستور، مجموع اولین عنصر و دومین عنصر (۱۲+۱۰) را در سومین عنصر آرایه قرار می‌دهد.

٤. برای تعیین تعداد عناصر آرایه از خاصیت Length استفاده می‌شود. دستورات زیر را ببینید:

```
Int[] a = new int [5];
Label1.Text = a.Length;
```

دستور اول، آرایه‌ای با ۵ عنصر تعریف می‌کند. دستور دوم، ۵ را در label۱ نمایش می‌دهد (تعداد عناصر آرایه a را با خاصیت Length تعیین می‌کند).

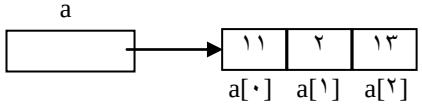
٥. در هنگام تعریف آرایه، عناصر آرایه از نوع عددی با صفر مقداردهی می‌شوند. عناصر آرایه از نوع boolean با false مقداردهی خواهند شد و عناصر آرایه از نوع ارجاع با **تهی**[۸] مقداردهی می‌گردند. ولی، می‌توان هنگام تعریف آرایه به آنها مقدار داد. برای این منظور باید به صورت زیر عمل نمایید:

;{مقادیر} [تعداد عناصر] نوع آرایه new = نام آرایه [ ] نوع آرایه

اکنون دستورات زیر را ببینید:

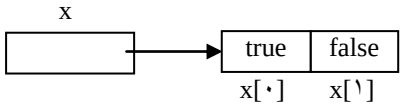
```
int[ ] a = new int [3] {11, 2, 13};
boolean[] x = new boolean [2] {true, false};
```

دستور اول، آرایه a را با سه عنصر تعریف کرده، مقادیر ۱۱، ۲ و ۱۳ را به عناصر آن تخصیص می‌دهد (شکل زیر):



۸- Null

دستور دوم، آرایه‌ای به نام x از نوع boolean تعریف کرده، مقادیر زیر را به آن تخصیص می‌دهد:



۶. برای تعریف آرایه چند بعدی باید به صورت زیر عمل کنید:

[طول بعد n و ... و طول بعد۲ و طول بعد۱] نوع آرایه new = نام آرایه [, ... ,] نوع آرایه

به عنوان مثال، دستورات زیر را ببینید:

```
int [ , ]   a = new  int [2,3] {{1,2,3},{4,5,6}};
int [ , , ] b = new  int [3,5,2];
```

دستور اول، آرایه‌ای به نام a با ۲ سطر و ۳ ستون تعریف می‌کند و مقادیر زیر را به آن تخصیص می‌دهد:

۱	۲	۳
٤	٥	٦

دستور دوم، آرایه‌ای به نام b با ۳ سطر، ۵ ستون و ارتفاع ۲ تعریف می‌کند و در خانه‌های آن مقدار صفر قرار می‌دهد. در ضمن برای دستیابی به عناصر آرایه چند بعدی از چند اندیس استفاده می‌شود. به عنوان مثال، برای بازیابی عناصر آرایه دو بعدی از دو اندیس استفاده می‌گردد. دستورات زیر را مشاهده کنید:

```
a [0,2] = 10;
a [1,0] = ++a [0,2];
```

دستور اول، مقدار عنصر سطر صفرام ستون دوم را برابر ۱۰ قرار می‌دهد و دستور دوم، ابتدا، عنصر سطر صفرام ستون دوم را یک واحد اضافه کرده، در عنصر سطر اول ستون صفرام قرار می‌دهد.

پیمایش عناصر آرایه

برای پیمایش عناصر آرایه می‌توان از حلقه‌های تکرار for، while، do while استفاده کرد. ولی استفاده از حلقه تکرار foreach، بهترین روش پیمایش آرایه‌ها و کلکسیون‌ها می‌باشد. ساختار foreach به صورت زیر به کار می‌رود:

foreach (نوع متغیر in نام آرایه یا نام کلکسیون)

{

; مجموعه دستورات بدنه حلقه

}

اکنون، دستور زیر را ببینید:

```
int a = new int [3] {1,15,18};
foreach (int i in a)
   listBox1.Items.Add (i.ToString());
```

دستور اول، آرایه‌ای به نام a با ۳ عنصر تعریف می‌کند و مقادیر ۱، ۱۵ و ۱۸ را به آنها تخصیص می‌دهـد و دستور دوم، هر یک از عناصر آرایه را در i قرار داده به listBox۱ اضافه می‌نماید.

مثال ۱-۱ برنامه‌ای که اعمال زیر را انجام می‌دهد:

۱. دو عدد صحیح را خوانده یکی از اعمال زیر را انجام می‌دهد:

- عدد اول را به توان عدد دوم می‌رساند (با عملگر +).
- عدد اول را در عدد دوم ضرب می‌کند (با عملگر +).
- عدد اول را بر عدد دوم تقسیم می‌کند (با عملگر -).

برای تعیین هر یک از این اعمال از یک دکمه RadioButton استفاده می‌کند.

۲. در هنگام دریافت عدد از ورود اطلاعاتی به جز ارقام جلوگیری می‌کند.

۳. فاکتوریل عدد کوچک‌تر را حساب می‌کند.

٤. ارقام اعداد بزرگ‌تر را در یک آرایه قرار داده سپس آنها را در یک ListBox قرار می‌دهد.

٥. کاربر می‌تواند اطلاعات ListBox را انتخاب کرده و اطلاعات انتخاب شده را بـه ListBox جدیـدی انتقـال دهد.

٦. کاربرمی‌تواند بین TextBoxها با کلیدهای ↑ , ↓ , ← حرکت کند.

مراحل طراحی و اجرا

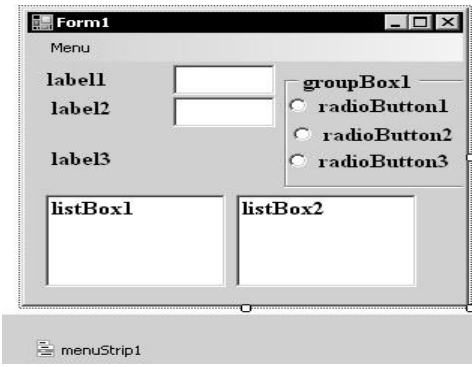
۱. با گزینه File/ New/ Project پروژه جدیدی ایجـاد کنیـد. در بخـش Name، ۱-۱ را وارد کـرده در بخش Templates، آیکن Windows Forms Application را انتخاب کرده دکمه OK را کلیک کنید.

۲. سه کنترل Label و دو کنترل TextBox به فرم اضافه کنید. کنتـرل‌هـای textBox۱ و textBox۲ بـه ترتیب برای خواندن عدد اول و عدد دوم به کار می‌روند.

۳. یک کنترل GroupBox به‌فرم اضافه کرده و سه‌کنترل RadioButton بـه‌کنتـرل GroupBox اضـافه نمایید.

٤. دو کنترل ListBox به فرم اضافه کنید.

٥. یک کنترل MenuStrip به فرم اضافه کنید.



٦. گزینه TypeHere را کلیک کـرده در داخـل آن Menu را تایپ کنید.

- به گزینه زیر Menu بروید و گزینـه Execute را تایپ کنید.
- به گزینه زیر Execute برویـد و گزینـه Fact را تایپ نمایید.
- در زیر گزینه Fact، گزینـه ListBox را اضـافه کنید.
- در زیر گزینه ListBox گزینه Move را تایپ کنید.
- در زیر گزینه Move، گزینه Exit را تایپ کنید.

۷. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Form۱_Load را به صورت زیر تایـپ کنید:

```
private void Form۱_Load(object sender, EventArgs e)
{
    radioButton۱.Text = "Pow";
    radioButton۲.Text = "Mul";
    radioButton۳.Text = "Div";
    label۱.Text = "Number۱";
    label۲.Text = "Number۲";
    label۳.Text = "";
    groupBox۱.Text = "Select one";
```

```
    listBox١.SelectionMode = SelectionMode.MultiSimple;
}
```

دستورات ۱ تا ۳، عنوان radioButtonها را تغییر می‌دهند. دستورات ٤ تا ٦ عنـوان Labelهـا را مقـداردهی می‌کنند. دستور هفتم، عنوان کنترل groupBox۱ را انتخاب می‌کنـد و دسـتور آخـر، listBox۱ را طـوری تغییـر می‌دهد که بتوان چندین گزینه را در آن انتخاب کرد.

۸. گزینه Execute در منوی Menu را کلیک مضاعف کرده، دستورات آن را به صورت زیر تغییر دهید:

```
private void execToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (textBox١.Text != "" && textBox٢.Text != "")
   {
      int n = Convert.ToInt٣٢(textBox١.Text);
      int m = Convert.ToInt٣٢(textBox٢.Text);
      int sum = ٠;
      if (radioButton١.Checked == true)
      {
         int k = n;
         for (int i = ١; i < m; i++)
         {
            sum = ٠;
            for (int j = ١; j <= n; j++)
                 sum += k;
            k = sum;
         }
         label٣.Text = "Pow is " + sum.ToString();
       }
       else  if (radioButton٢.Checked == true)
       {
          sum = ٠;
          int i =١;
          while(i <= m)
          {
             sum += n ;
             i++;
```

```
            }
            label۳.Text = "Mul is " + sum.ToString();
        }
        else if (radioButton۳.Checked == true)
        {
          int i = ۰;
          while (m <= n)
          {
             n -= m;
             i++;
          }
          label۳.Text = "Div is " + i.ToString();
        }
      }
}
```

این دستورات، ابتدا چک می‌کنند که مقادیر textBox۱ و textBox۲ خالی نباشند. در این صورت، آنها را به عدد تبدیل کرده، در متغیرهای m و n قرار می‌دهد. در ادامه چک می‌نماید آیا radioButton۱ انتخاب شده است یا خیر؟ اگر انتخاب شده باشد، بخش بعدی الگوریتم، n را به توان m رسانده نمایش می‌دهد. اگر کنترل radioButton۱ انتخاب شده نباشد، کنترل radioButton۲ را بررسی می‌کند، اگر انتخاب شده باشد، عدد n را در m ضرب کرده (حلقه تکرار) نمایش می‌دهد. اگر radioButton۲ انتخاب نگردید، کنترل radioButton۳ را بررسی می‌نماید و در صورتی که کاربر آن را انتخاب کرده باشد، با استفاده از یک حلقه تکرار باقیمانده عدد اول بر عدد دوم را محاسبه نموده، نمایش می‌دهد.

۹. در منوی Menu، گزینه Fact را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تایپ کنید:

```
private void factToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (textBox۱.Text != "" && textBox۲.Text != "")
    {
       int n = Convert.ToInt۳۲(textBox۱.Text);
       int m = Convert.ToInt۳۲(textBox۲.Text);
       long fact = ۱;
       int x;
       if (m < n)
          x = m;
```

```
        else
           x = n;
        for (int i= ۲; i <= x; i++)
           fact *= i;
        label۳.Text = "Fact is " + fact.ToString();
     }
}
```

این دستورات، اگر مقادیر textBox۱ و textBox۲ خالی نباشند، اعداد موجود در textBoxها را در متغیرهای m و n قرار می‌دهند، عدد کوچک‌تر را پیدا کرده، فاکتوریل آن را محاسبه می‌کنند و نمایش می‌دهند.

۱۰. گزینهListBox از منویMenu را کلیک مضاعف کرده، دستورات رویدادClick آن را به صورت زیر تغییر دهید:

```
private void listBoxToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (textBox۱.Text != "" && textBox۲.Text != "")
   {
      int n = Convert.ToInt۳۲(textBox۱.Text);
      int m = Convert.ToInt۳۲(textBox۲.Text);
      int x, i =۰;
      int[] a = new int[۱۲];
      if (m > n)
         x = m;
      else
         x = n;
      while (x > ۰)
      {
        a[i] = x % ۱۰;
        x /= ۱۰;
        i++;
      }
      listBox۱.Items.Clear();
      for (int j = i - ۱; j >= ۰; j--)
         listBox۱.Items.Add(a[j].ToString());
```

```
    }
}
```

این دستورات، ابتدا چک می‌کنند آیا کنترل‌های textBox۱ و textBox۲ دارای مقدار هستند. اگر دارای مقدار باشند، عدد بزرگ‌تر را پیدا می‌نمایند و توسط حلقه تکرار while، ارقام آن را جدا کرده به آرایه a انتقال می‌دهند. دستورات بعدی، اطلاعات آرایه را از آخرین رقم به اولین رقم (آخرین عنصر آرایه به اولین عنصر آرایه) در listBox۱ اضافه می‌کنند.

۱۱. گزینه Move در منوی Menu را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تایپ کنید:

```
private void moveToolStripMenuItem_Click(object sender, EventArgs e)
{
   listBox2.Items.Clear();
   foreach (string   i in  listBox1.SelectedItems)
   {
      listBox2.Items.Add(i.ToString());
   }
}
```

این دستورات عناصر انتخاب شده در کنترل listBox۱ را در کنترل listBox۲ کپی می‌نمایند.

۱۲. گزینه Exit از منوی Menu را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تایپ کنید:

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
   Close();
}
```

۱۳. رویداد KeyPress کنترل textBox۱ را به برنامه اضافه کنید. برای این منظور مراحل زیر را انجام دهید:

- کنترل textBox۱ را کلیک کنید تا انتخاب شود.
- در پنجره Properties دکمه Events را کلیک کنید. در لیست رویدادهایی که ظاهر می‌شوند، رویداد KeyPress را پیدا کرده، کلیک مضاعف کرده و دستورات آن را به صورت زیر تغییر دهید:

```
private void textBox۱_KeyPress(object sender, KeyPressEventArgs e)
```

```
{
   if (e.KeyChar <= '۹' && e.KeyChar >= '۰')
      e.Handled = false;
   else
      e.Handled = true;
}
```

این دستورات موجب می‌شوند تا textBox۱ فقط ارقام را بپذیرد.

۱٤. مانند مرحله ۱۳، رویداد KeyPress را برای کنترل textBox۲ اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
private void textBox۲_KeyPress(object sender, KeyPressEventArgs e)
{
   if (e.KeyChar <= '۹' && e.KeyChar >= '۰')
       e.Handled = false;
   else
       e.Handled = true;
}
```

این دستورات، از ورود اطلاعات غیر ارقام در کنترل textBox۲ جلوگیری می‌کنند.

۱۵. رویداد KeyDown را برای کنترل textBox۱ به برنامه اضافه کرده، دستورات رویداد KeyDown آن را به صورت زیر تغییر دهید:

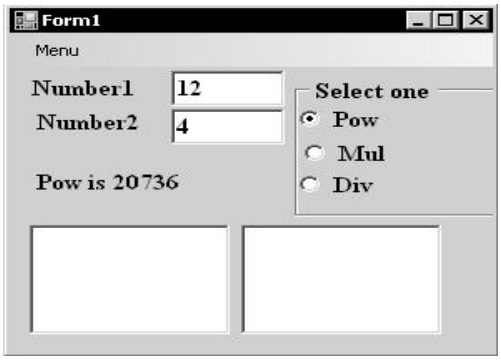
```
private void textBox۱_KeyDown(object sender, KeyEventArgs e)
{
   if (e.KeyValue == ۱۳ || e.KeyValue == ۳۸ || e.KeyValue == ٤۰)
         textBox۲.Focus();
}
```

۱٦. رویداد KeyDown را برای textBox۲ اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

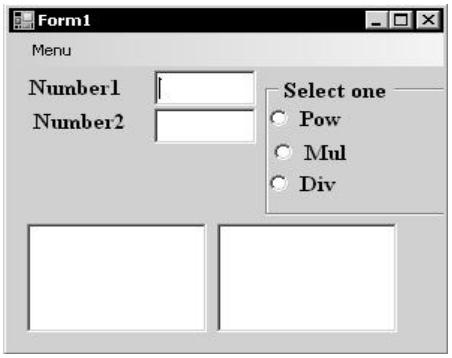
```
private void textBox۲_KeyDown(object sender, KeyEventArgs e)
{
   if (e.KeyValue == ۱۳ || e.KeyValue== ۳۸ || e.KeyValue == ٤۰)
       textBox۱.Focus();
```

}

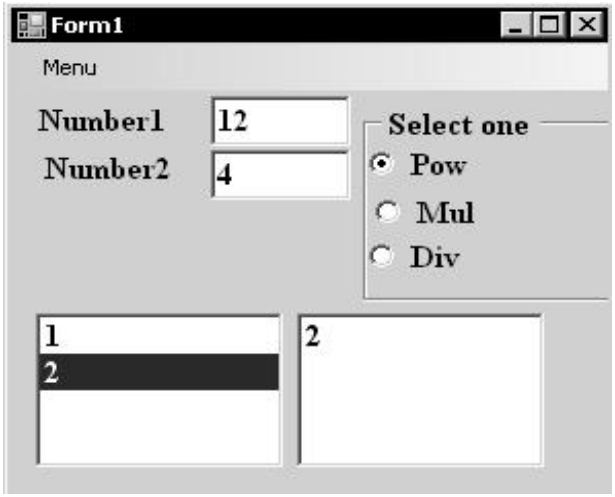
۱۷. پروژه را ذخیره و اجرا کنید. اکنون شکل ۳-۱ ظاهر می‌شود. گزینه Pow را انتخاب کرده به ترتیب اعداد ۱۲ و ٤ را وارد نمایید و در منوی Menu، گزینه Execute را اجرا کنید تا شکل ٤-۱ ظاهر شود. گزینه listBox را از منوی Menu اجرا کنید. سپس، عدد ۲ را انتخاب کرده، گزینه Move از منوی Menu را انتخاب کنید تا خروجی به شکل ۵-۱ ظاهر گردد. برای خروج از برنامه گزینه Exit از منوی Menu را اجرا نمایید.



شکل ۳-۱ نمونه خروجی مثال۱-۱.



شکل ۴-۱ نمونه خروجی محاسبه توان.



شکل ۵-۱ نمونه خروجی کپی گزینه از یک listBox به listBox دیگر.

۱۳–۱. کلاس‌ها و اشیاء

کلاس، الگویی برای اشیایی با ویژگی‌ها و صفات یکسان می‌باشد. به عنوان مثال، کلاس، **حیوانات** را در نظر بگیرید که ویژگی تمام حیوانات را داشته باشد. بعضی از این ویژگی‌ها عبارتند از:

۱. غذا می‌خورند.
۲. می‌خوابند.
۳. بیدار می‌شوند.
۴. می‌میرند.
۵. حرکت می‌کنند.
۶. و غیره

این کلاس، به عنوان یک الگو برای حیوانات است. کلاس حیوانات برای تعریف حیوانات مختلف به کار می‌رود که هر حیوان می‌تواند ویژگی‌های خاصی داشته باشد:

۱. خزنده است یا پرنده.
۲. تخم‌گذار است یا پستاندار.
۳. اهلی است یا وحشی.
۴. و غیره

همین‌طور که در این کلاس دیدید، گرچه حیوانات با هم فرق دارند، ولی صفات مشترکی دارند که نشان می‌دهد به نحوی با هم ارتباط دارند. برای استفاده از کلاس باید دو کار انجام گردد:

۱. تعریف کلاس
۲. نمونه‌سازی از کلاس

۱–۱۳–۱. تعریف کلاس

برای تعریف کلاس از واژه class به صورت زیر استفاده می‌شود:

```
نام کلاس    class    سطح دستیابی کلاس
{
اعضای کلاس
}
```

سطح دستیابی کلاس، تعیین می‌کند کلاس تعریف شده را کجا می‌توانید استفاده کنید. سطح دستیابی می‌تواند public یا internal باشد. واژه public کلاس را طوری تعریف می‌کند که در خارج از فضای نامی که در آن تعریف شده است، قابل استفاده باشد و internal، کلاس را طوری تعریف می‌کند که فقط در فضای نام تعریف شده قابل دستیابی باشد. اعضای کلاس را در ادامه می‌آموزیم.

۲-۱۳-۱. نمونه‌سازی کلاس

برای استفاده از کلاس باید اشیایی از نوع آن ایجاد کرد. نمونه‌سازی کلاس به صورت زیر می‌باشد:

نام کلاس نام نمونه = new نام کلاس ( ) ;

۳-۱۳-۱. اعضای کلاس

یکی از بخش‌های بسیار مهم کلاس، اعضای آن است. کلاس می‌تواند اعضای زیر را داشته باشد:

۱. ثوابت **۲.** فیلدها **۳.** خواص **٤.** متدها

٥. سازنده‌ها **٦.** مخرب‌ها **۷.** و ...

ثوابت: مقادیری هستند که تغییر نمی‌یابند. تعریف ثوابت به صورت زیر است:

سطح دستیابی const نوع نام ثابت = مقدار ;

به عنوان مثال، دستور زیر را مشاهده کنید:

```
private const int x = ۱۰;
```

این دستور ثابت x از نوع int را با دستیابی خصوصی (در خارج از کلاس قابل دستیابی نمی‌باشد) تعریف می‌کند.

فیلدها: همانند متغیر معمولی برای ذخیره کردن داده‌ها به کار می‌روند و به صورت زیر تعریف می‌شوند:

سطح دستیابی نوع نام [مقدار=] ;

به عنوان مثال، دستورات زیر را ببینید:

```
public int x;
private float y;
```

دستور اول، فیلدی به نام x را با دستیابی عمومی تعریف می‌کند (در خارج از کلاس می‌توان به آن دسترسی داشت) و دستور دوم، فیلدی به نام y از نوع float را با دستیابی خصوصی تعریف می‌کند.

خواص: متدهایی هستند که داده‌های فیلدها را مقداردهی کرده یا مقدار آنها را بازیابی می‌نمایند. برای مقداردهی به خاصیت از **واژه set** و برای بازیابی مقدار خاصیت از **واژه get** استفاده می‌گردد. به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
public   float   Y
```

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

این دستور، خاصیتی به نام Y را تعریف می‌کند که مقدار فیلد y را بازیابی کرده یا آن را مقدار می‌دهد.

متدها: مجموعه دستوراتی هستند که عمل خاصی را به روی کلاس انجام می‌دهند. برای استفاده از متدها دو کار باید انجام شود:

۱. تعریف متد **۲.** فراخوانی متد

تعریف متد: به صورت زیر انجام می‌شود:

(لیست پارامترهای مجازی) نام متد نوع مقدار برگشتی متد سطح دستیابی متد

{

; دستورات بدنه متد

}

سطح دستیابی متد می‌تواند public، private یا غیره باشد که با این واژه‌ها آشنا شدید. پارامترها، مقادیری هستند که باید به متد ارسال شوند.

فراخوانی متد: برای استفاده از متد تعریف آن کافی نیست، بلکه باید آن را اجرا کرد. برای اجرای متد آن را باید به صورت زیر فراخوانی نمود:

; (لیست پارامترهای واقعی) نام متد

همان‌طور که دیدید، وقتی متدی را فراخوانی می‌کنید، می‌توانید مقداری را به متد فراخوان برگردانید. نوع این مقدار در تعریف متد **با نوع مقدار برگشتی متد** مشخص می‌گردد. اگر به جای **نوع متد** از کلمه void استفاده کنید، متد هیچ مقداری را از طریق **نام متد** به متد فراخوان برگشت نمی‌دهد. ولی ممکن است از طریق پارامترها مقادیری را برگشت دهد که در ادامه به این موضوع می‌پردازیم.

روش‌های ارسال پارامترها

دو روش برای ارسال پارامترها وجود دارد که عبارتند از:

۱. ارسال با مقدار (pass by value)

۲. ارسال با ارجاع (pass by reference)

- **ارسال با مقدار**، یک کپی از مقدار پارامترهای واقعی را در پارامتر مجازی قرار می‌دهد. یعنی، پارامترهای مجازی و واقعی دو مکان مستقل از حافظه هستند که تغییر پارامترهای مجازی (در متد فراخوانی شده) هیچ تأثیری بر پارامترهای واقعی (در متد فراخوان) نمی‌گذارد. در این روش، ارتباط بین پارامترهای واقعی و مجازی یک طرفه است.
- **ارسال با ارجاع**، آدرس پارامترهای واقعی را در پارامترهای مجازی قرار می‌دهد. یعنی، پارامترهای واقعی و مجازی به یک خانه از حافظه اشاره می‌کنند و هر تغییری در پارامتر مجازی در پارامتر واقعی نظیر آن اعمال خواهد شد. در این روش، ارتباط بین پارامترهای واقعی و مجازی دو طرفه است.

برای ارسال با ارجاع، هم در هنگام تعریف پارامترهای مجازی و هم در هنگام فراخوانی با پارامترهای واقعی از یکی از واژه‌های **ref** یا **out** استفاده کنید. اگر از واژه **ref** استفاده کنید، حتماً باید به پارامتر واقعی، قبل از فراخوانی، مقدار اولیه تخصیص دهید. ولی، اگر از واژه **out** استفاده نمایید، نیازی به این کار نیست.

سازنده‌ها: وقتی نمونه‌ای از کلاس را ایجاد می‌کنید، سازنده کلاس فراخوانی می‌شود. با استفاده از سازنده می‌توانید در هنگام ایجاد نمونه، اعضای داده‌ای کلاس را مقدار اولیه دهید. یک کلاس می‌تواند هیچ سازنده، یک سازنده یا چند سازنده داشته باشد. اگر کلاسی سازنده نداشته باشد، از سازنده پیش‌فرض خودش استفاده می‌کند. **سازنده**، متدی است که نام آن هم نام کلاس است و دارای نوع نمی‌باشد (حتی نوع void). به عنوان مثال، دستورات زیر را ببینید:

```
public  class  test
{
   private const int x = ۱۰;
   private int a;
   public int A;
   {
      get {    return a;      }
      set {    a = value;     }
   }
   public  test  (int p)  {     a = p;       }
   public  test( )  {  a = ۰;  }
   public void inca ( ) {  a ++; }
}
```

این دستورات، کلاسی به نام test را تعریف می‌کنند که دارای ویژگی‌های زیر است:

- یک مقدار ثابت به نام x با مقدار ۱۰ دارد.
- یک فیلدی به نام a دارد.

- خاصیتی به نام A دارد که برای بازیابی و مقداردهی به فیلد a به کار می‌رود.
- متدی به نام inca دارد که مقدار فیلد a را یک واحد اضافه می‌کند.
- سازنده‌ای به نام test دارد که یک پارامتر را گرفته، مقدار آن را در فیلد a قرار می‌دهد.
- سازنده دیگری بدون پارامتر دارد که مقدار فیلد a را برابر صفر قرار می‌دهد.

اکنون دستور زیر را در نظر بگیرید:

```
test t۱ = new test (۱۲);
```

این دستور، نمونه‌ای به نام t۱ از نوع test ایجاد می‌کند که مقدار فیلد a آن برابر ۱۲ خواهد شد.

برای دستیابی به اعضای کلاس باید به صورت زیر عمل کنید:

نام عضو.نام نمونه

به عنوان مثال، دستورات زیر را ببینید:

```
t۱.A = ۱۲;
t۱.inca( );
```

دستور اول، فیلد a نمونه t۱ را برابر ۱۲ قرار می‌دهد و دستور دوم، متد inca را بر روی نمونه t۱ فراخوانی می‌کند. یعنی، مقدار فیلد a را یک واحد اضافه می‌کند.

اعضای استاتیک: از یک کلاس می‌توان چندین نمونه را ایجاد کرد و هر شیء دارای یک کپی از متغیرهای خودش است. به عنوان مثال، اگر کلاس دارای یک فیلدی به نام a باشد و ۱۰۰ نمونه از آن ایجاد کنید، هر یک از اشیا کلاس (۱۰۰ نمونه) یک کپی از فیلد a را دارند. اما گاهی لازم است، فیلدی را داشته باشید که بین نمونه‌ها مشترک باشد و به جای این که برای هر نمونه فضایی به آن اختصاص دهد، فقط یک فضا به آن اختصاص داده و تمام اشیا به آن دسترسی داشته باشند. برای این منظور می‌توانید از واژه static (همان فیلدهای استاتیک) استفاده کنید. این فیلدها، معمولاً برای تبادل اطلاعات بین فرم‌های مختلف به کار می‌روند. فیلدهای static به صورت زیر تعریف می‌شوند:

سطح دستیابی static نوع فیلد نام فیلد [=مقدار] ;

به عنوان مثال، دستور زیر را ببینید:

```
public static int s = ۱۲;
```

این دستور، فیلد s را از نوع static تعریف کرده و مقدار ۱۲ را به آن تخصیص می‌دهد.

مثال۲-۱ برنامه‌ای که کلاس اعداد کسری را ایجاد می‌کند و اعمال زیر را بر روی آن انجام می‌دهد:

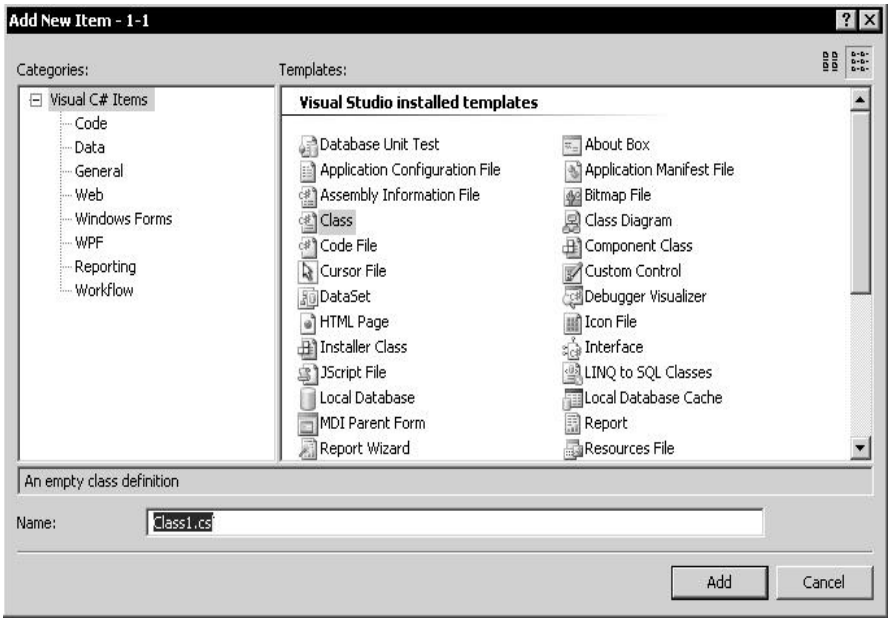
- سازنده‌ای دارد که از مخرج صفر جلوگیری می‌کند.
- متدهایی برای جمع، تفریق، ضرب و تقسیم کسرها دارد.
- متد ToString را مجدداً پیاده‌سازی می‌کند تا کسر را نمایش دهد.

مراحل طراحی و اجرا

۱. پروژه جدیدی ایجاد کنید.

۲. گزینه ... Project/ Add Class را اجرا کنید تا پنجره Add New Item ظاهر شود. (شکل۶-۱)

۳. در بخش Name، نام کلاس را Fraction.cs وارد کرده، دکمه Add را کلیک کنید.



شکل۶-۱ پنجره Add New Item.

اکنون کلاس جدید ایجاد شده، دستورات آن به صورت زیر ظاهر می‌شوند:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace _۱_۲

{
```

```
    class Fraction
    {
    }
}
```

دستورات کلاس Fraction را به صورت زیر تغییر دهید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace _۱_۲
{
    class Fraction
    {
        private int x;
        private int y;
        public Fraction()
        {
            x = ۰;
            y = ۱;
        }
        public Fraction(int a, int b)
        {
          if (b == ۰)
          {
              x = ۰;
              y = ۱;
          }
          else
          {
              x = a;
              y = b;
          }
        }
        public int X
        {
            get      {     return x;      }
            set      {     x = value;     }
         }
         public int Y
         {
            get      {     return y;      }
```

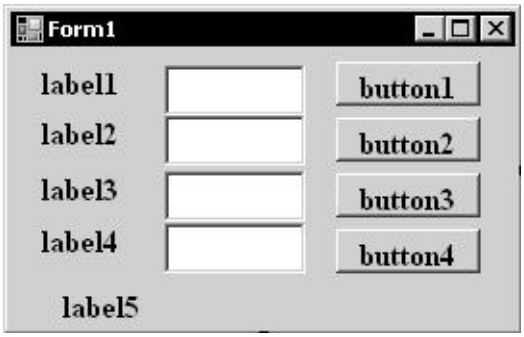
```
            set
            {
               if ( value == ‘)

                   y=\;
               else
                   y = value;
            }
         }
         public Fraction Add(Fraction f)
         {
            Fraction temp= new Fraction();
            temp.x = x * f.y + y * f.x;
            temp.y = y * f.y;
            return temp;
         }
         public Fraction Sub(Fraction f)
         {
            Fraction temp= new Fraction();
            temp.x = x * f.y - y * f.x;
            temp.y = y * f.y;
            return temp;
         }
         public Fraction Mul(Fraction f)
         {
            Fraction temp=new Fraction();
           temp.x = x * f.x;
           temp.y = y * f.y;
           return temp;
         }
         public Fraction Div(Fraction f)
         {
            Fraction temp = new Fraction();
            temp.x = x * f.y;
            temp.y = y * f.x;
            return temp;
          }
      public   override string  ToString()
      {
        string s = "";
        s = s + x.ToString() + " / " + y.ToString();
      return s;
      }
    }
}
```

این کلاس دارای اعضای زیر است:

- **فیلد x**، مقدار صورت کسر را نگهداری می‌کند.
- **فیلد y**، مقدار مخرج کسر را نگهداری می‌کند.
- **سازنده Fraction( )**، صورت را برابر صفر و مخرج را برابر ۱ قرار می‌دهد.
- **سازنده Fraction (int a, int b)**، اگر مخرج کسر برابر صفر باشد، صورت کسر را صفر و مخرج کسر را ۱ قرار می‌دهد. وگرنه، صورت کسر را برابر پارامتر a و مخرج کسر را برابر پارامتر b قرار می‌دهد.
- **خاصیت X**، برای مقداردهی و بازیابی مقدار فیلد x به کار می‌رود.
- **خاصیت Y**، برای مقداردهی و بازیابی مقدار فیلد y به کار می‌رود.
- **متد Add**، دو عدد کسری را با هم جمع کرده (کسر اول، از طریق ضمنی ارسال می‌شود و کسر دوم از طریق پارامتر f ارسال خواهد شد) نتیجه را در temp قرار می‌دهد و برمی‌گرداند.
- متد **Sub**، دو کسر را از هم کم کرده، از طریق کسر temp برمی‌گرداند.
- متد **Mul**، دو کسر را در هم ضرب کرده از طریق کسر temp برمی‌گرداند.
- متد **Div**، دو کسر را بر هم تقسیم کرده از طریق کسر temp برمی‌گرداند.
- متد **ToString**، متد ()ToString را مجدداً پیاده‌سازی می‌کند تا رشته‌ای کسری را برگرداند.

۴. ۵ کنترل Label، ٤ کنترل TextBox و ٤ کنترل Button به فرم اضافه کنید.



۵. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Load آن را به صورت زیر تغییر دهید:

```
private void Form۱_Load(object sender, EventArgs e)
{
   label۱.AutoSize = true;
   label۲.AutoSize = true;
```

```
    label٣.AutoSize = true;
    label٤.AutoSize = true;
    label٥.AutoSize = true;
    label١.Text = "Enter x١:";
    label٢.Text = "Enter y١:";
    label٣.Text = "Enter x٢:";
    label٤.Text = "Enter y٢:";
    label٥.Text = "";
    button١.Text = "Add";
    button٢.Text = "Sub";
    button٣.Text = "Mul";
    button٤.Text = "Div";
}
```

این دستورات، خواص کنترل‌های روی فرم را مقداردهی می‌کنند.

٦. دکمه button١ را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
private void button١_Click(object sender, EventArgs e)
{
    Fraction f١ = new Fraction();
    f١.X = Convert.ToInt١٦(textBox١.Text);
    f١.Y = Convert.ToInt١٦(textBox٢.Text);
    Fraction f٢ = new Fraction();
    f٢.X = Convert.ToInt١٦(textBox٣.Text);
    f٢.Y = Convert.ToInt١٦(textBox٤.Text);
    Fraction result = new Fraction();
    result = f١.Add(f٢);
    label٥.Text = "Result is " + result.ToString();
}
```

دستور۱، شیء f۱ را از نوع Fraction تعریف می‌کند. دستورات ۲ و ۳، فیلدهای x و y را از طریق خواص X و Y مقداردهی می‌کنند. دستور٤، شیء f۲ را از نوع Fraction ایجاد می‌کند. دستورات ٥ و ٦، فیلدهای x و y را از طریق خواص X و Y مقداردهی می‌کنند. دستور۷، شیء result را ایجاد می‌کند. دستور۸، متد Add را بر روی اشیا f۱ و f۲ اجرا کرده، نتیجه را در شیء result قرار می‌دهد و دستور آخر، متد ToString را بر روی شیء result اجرا کرده، نتیجه را برمی‌گرداند.

۷. دکمه button۲ را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
private void button۲_Click(object sender, EventArgs e)
{
   Fraction f۱ = new Fraction(Convert.ToInt۱٦(textBox۱.Text),
   Convert.ToInt۱٦(textBox۲.Text));
   Fraction f۲ = new Fraction(Convert.ToInt۱٦(textBox۳.Text),
   Convert.ToInt۱٦(textBox٤.Text));
    Fraction result = new Fraction();
    result = f۱.Sub(f۲);
    label٥.Text = "Result is " + result.ToString();
}
```

دو دستور اول از طریق سازنده Fraction اشیا f۱ و f۲ را ایجاد می‌کنند. دستور سوم، شیء result را ایجاد می‌کند. دستور چهارم، متد Sub را به روی شیء f۱ و f۲ اجرا کرده نتیجه را در result قرار می‌دهد و دستور آخر، متد ToString را به روی شیء result اجرا کرده، نتیجه تفریق را در Label٥ نمایش می‌دهد.

۸. دکمه button۳ را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
private void button۳_Click(object sender, EventArgs e)
{
   Fraction f۱ = new Fraction(Convert.ToInt۱٦(textBox۱.Text),
   Convert.ToInt۱٦(textBox۲.Text));
   Fraction f۲ = new Fraction(Convert.ToInt۱٦(textBox۳.Text),
   Convert.ToInt۱٦(textBox٤.Text));
   Fraction result = new Fraction();
```

```
    result = f١.Mul(f٢);
    label٥.Text = "Result is " + result.ToString();
}
```

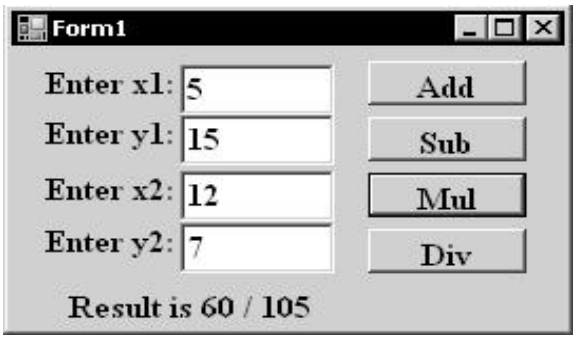
این متد، برای ضرب دو کسر به کار می‌رود. و مانند متد Sub عمل می‌کند، با این تفاوت که به جای فراخوانی متد Sub، متد Mul را فراخوانی می‌کند.

۹. دکمه button٤ را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
private void button٤_Click(object sender, EventArgs e)
{
    Fraction f١ = new Fraction(Convert.ToInt١٦(textBox١.Text),
    Convert.ToInt١٦(textBox٢.Text));
    Fraction f٢ = new Fraction(Convert.ToInt١٦(textBox٣.Text),
     Convert.ToInt١٦(textBox٤.Text));
    Fraction result = new Fraction();
    result = f١.Div(f٢);
    label٥.Text = "Result is " + result.ToString();
}
```

این متد، مانند متد Mul می‌باشد. با این تفاوت که دو کسر را بر هم تقسیم کرده، نمایش می‌دهد.

۱۰. پروژه را ذخیره و اجرا کنید. در جلوی Enter x۱، عدد ۵، در جلوی Enter y۱، عدد ۱۵، در جلوی Enter x۲، عدد ۱۲ و جلوی Enter y۲ عدد ۷ را وارد کرده، دکمه Mul را کلیک کنید تا خروجی را به شکل ۷-۱ ببینید.



شکل۷-۱ خروجی مثال۲-۱.

فصل ۲ مبانی بانک اطلاعات SQL Server

امروزه سازمان‌ها، مؤسسات، ادارات و شرکت‌ها با حجم عظیمی از داده‌ها سر و کار دارند. به عنوان مثال، فرض کنید بخواهید اطلاعات مربوط به مکالمات شرکت مخابرات یکی از استان‌ها را نگهداری کنید. به طوری که در یک سال حدود ۱۵ میلیارد رکورد جمع‌آوری می‌گردد. نگهداری، پردازش و بازیابی این حجم اطلاعات از طریق فایل‌های معمولی زمان‌بر است. برای جلوگیری از تکرار بی‌مورد داده‌ها (افزونگی داده‌ها)، ایجاد سازگاری بین گزارش‌ها (جلوگیری از بی نظمی) و صرفه‌جویی در میزان حافظه، به کارگیری بانک اطلاعات به صورت یک ضرورت درآمده است. یعنی، بدون استفاده از بانک اطلاعات نمی‌توان اطلاعات مربوط به مکالمات تلفن ثابت یک استان را نگهداری و ذخیره کرد. از طرف دیگر، اکثر برنامه‌های کاربردی که با داده‌ها سر و کار دارند، داده‌ها را در بانک اطلاعات ذخیره می‌نمایند و از طریق بانک اطلاعات آن را پردازش می‌کنند.

بانک‌های اطلاعات متعددی وجود دارند که از جمله می‌توان Access، SQL Server، Oracle، DB۲ و MySQL را نام برد. هر یک از این بانک‌های اطلاعات کاربرد خاصی دارند. در بین این بانک‌ها، بانک اطلاعات SQL Server از محبوبیت خاصی برخوردار است. زیرا، حدود ۷۰ درصد از کاربران دنیا از این بانک اطلاعات استفاده می‌کنند. به همین خاطر، در این فصل ابتدا با مفاهیم بانک اطلاعات و دلایل استفاده از آن آشنا خواهیم شد. سپس، یک بانک اطلاعات نمونه را مثال می‌زنیم و در ادامه کتاب، این بانک اطلاعات را ایجاد کرده، اعمال مختلف را بر روی آن انجام می‌دهیم.

۱-۲. تعریف سیستم مدیریت بانک اطلاعات

سیستم مدیریت **بانک اطلاعات**، مکانیزیم نگهداری رکوردها[۹] است. یعنی، بانک اطلاعات مخزنی برای نگهداری از داده‌ها است که کاربران آن می‌توانند اعمال زیر را انجام دهند:

۹- **Records** ۲- **Query**

۱. افزودن جداول خالی به بانک اطلاعات

۲. افزودن رکوردهایی به جداول بانک اطلاعات

۳. تغییر ساختار جداول

٤. حذف رکوردهای بانک اطلاعات

۵. تغییر داده‌های موجود دربانک اطلاعات

٦. اجرای پرس‌وجو[۱۰] بر روی جداول

به عبارت ساده‌تر، سیستم مدیریت بانک اطلاعات، سیستمی کامپیوتری است که هدف آن ذخیره و بازیابی داده‌ها می‌باشد. بانک اطلاعات داده را پردازش نموده به اطلاعات تبدیل کرده، آن را بازیابی می‌نماید.

۱-۱-۲. دلایل استفاده از بانک اطلاعات

شاید از خودتان بپرسید، بانک اطلاعات چه مزایایی دارد؟ پاسخ به این سوال به مواردی از قبیل اندازه سیستم، تعداد کاربران سیستم و غیره بستگی دارد. هرچه اندازه سیستم بزرگ‌تر شود و تعداد کاربران بیشتر گردد، ضرورت به کارگیری بانک اطلاعات بیشتر خواهد شد. برای بیان مزایای بانک اطلاعات مثال زیر را در نظر بگیرید:

فرض کنید برای فروشگاه ساده‌ای بانک اطلاعات طراحی می‌کنید. این بانک اطلاعات بسیار کوچک و ساده است و شاید امتیازات استفاده از بانک اطلاعات به چشم نیاید. امّا همین بانک اطلاعات را برای فروشگاه زنجیره‌ای بزرگ در نظر بگیرید که دارای انبارهای زیادی است و موجودی انبارها به سرعت تغییر می‌کند. امتیازات سیستم بانک اطلاعات نسبت به سیستم سنتی که رکوردها بر روی کاغذ نگهداری می‌شوند، در این‌گونه موارد، بیشتر به چشم می‌آیند. برخی از مزایای بانک اطلاعات در زیر آمده‌اند:

☒ **فشردگی**: چون داده‌های بانک اطلاعات دارای ساختار هستند، نیازی به نگهداری فایل‌های متنی حجیم نیست و از ورود داده‌های نامنظم (بدون ساختار) جلوگیری می‌کند. بنابراین، باعث فشرده‌سازی اطلاعات می‌گردد.

☒ **سرعت:** سیستم می‌تواند سریع‌تر از انسان داده‌ها را بازیابی و به هنگام کند. مخصوصاً سیستم توزیع شده باشد (مانند فروشگاه زنجیره‌ای)، پاسخ‌گویی به درخواست‌های **سراسری** و موردی بسیار سریع‌تر انجام می‌گردد.

☒ **بودجه کمتر:** خیلی از یکنواختی‌ها در نگهداری فایل‌ها به روش دستی و سنتی که به فضای زیادی نیاز دارند، حذف خواهند شد و دیگر نیازی به ساختمان‌های بزرگ و کارمندان زیادی برای نگهداری و پردازش اطلاعات نمی‌باشد.

☒ **دسترسی:** در هر زمان اطلاعات دقیق و به هنگام شده در اختیار قرار می‌گیرند. زیرا، اطلاعات به صورت مجتمع و یک‌پارچه نگهداری می‌شوند.

☒ **حفاظت:** داده‌ها می‌توانند در مقابل دستیابی غیرقانونی و غیرمجاز حفظ شوند. زیرا، اطلاعات در یک نقطه نگهداری می‌گردند. بنابراین، می‌توان از طریق فایروال، تعریف حساب کاربری و کلمه عبور از ورود افراد غیرمجاز جلوگیری کرد.

البته این مزایا در محیط چند کاربره که بانک‌های اطلاعات بزرگ و پیچیده باشند، چشمگیرتر است. اما، یک امتیاز ویژه در چنین محیطی وجود دارد و آن عبارت است از: سیستم بانک اطلاعات ایجاد می‌شود تا مؤسسه بر روی داده‌هایش **کنترل مرکزی** داشته باشد (این موضوع از اهمیت ویژه‌ای برخوردار است). این وضعیت با وضعیتی که در مؤسسات بدون بانک اطلاعات کار می‌کنند، متفاوت است. در مؤسسات فاقد بانک اطلاعات، هر برنامه کاربردی فایل‌های خاص خودشان را دارند، گاهی نیز نوارها و دیسک‌های مخصوص به خود دارند. بنابراین، داده‌ها پراکنده‌اند و کنترل بر روی داده‌ها با روش سیستماتیک دشوار است.

۲-۱-۲. طراحی بانک اطلاعاتی

بانک‌های اطلاعاتی مختلفی وجود دارند که مهم‌ترین آنها بانک اطلاعات رابطه‌ای است. اطلاعات در بانک اطلاعات رابطه‌ای، بین جداول مختلف توزیع می‌گردند تا ذخیره‌سازی و بازیابی اطلاعات بهینه‌تر شود. یعنی، از افزونگی داده‌ها جلوگیری می‌گردد، بی‌نظمی را کاهش می‌دهد و داده‌های تهی را نیز کاهش خواهد داد. این زمانی اتفاق می‌افتد که بانک اطلاعات خوب طراحی گردد. بنابراین، هرچه بانک اطلاعات بهتر طراحی شود، ابزار مهمی برای مدیریت بر اطلاعات شخصی، تجاری یا اداری است. ولی، چنانچه بانک اطلاعات بد طراحی گردد، ارزش چندانی نخواهد داشت. پس، هرچه وقت بیشتری برای طراحی و تحلیل داده‌ها صورت گیرد، نتیجه بهتری بدست می‌آید. زمانی که طراحی کامل بررسی گردید، به راحتی می‌توان بانک اطلاعات را ایجاد نمود.

فرآیند طراحی با تحلیل کارهایی شروع می‌گردد که برای بانک اطلاعات نیاز داریم. در این فرآیند، ابتدا باید مشخص کنید سیستم چه کاری باید انجام دهد. با کاربران مصاحبه کرده تا به خواسته‌های آن‌ها پی ببرید. توجه

داشته باشید که فرآیند طراحی، فرآیندی تکراری است. وقتی کاربران می‌خواهند از سیستم جدید استفاده کنند، راجع به ویژگی‌های آن فکر می‌کنند، مثل فرم‌های ورود داده‌ها، پرس‌وجوهای خاص، و فیلدهای محاسباتی.

از طرف دیگر، طراحی باید در یک نقطه خاتمه یابد و توسعه بانک اطلاعات شروع گردد. در این صورت، خواسته‌های جدید سیستم را می‌توانید در نسخه‌های بعدی سیستم منظور کنید. فرآیند طراحی بانک اطلاعات را می‌توان به مراحل زیر تقسیم کرد که هر مرحله دارای هدف خاصی است:

۱. تعیین خواسته‌های کاربران: در این مرحله، با کاربران مصاحبه می‌گردد. فرم کاربران بررسی می‌شود تا خواسته‌های آنها از بانک اطلاعات مشخص گردد.

۲. توزیع داده‌ها در جداول: یکی از مهم‌ترین بخش‌های طراحی، توزیع داده‌های جداول است. زیرا، چنانچه طراحی جداول خوب باشد، از تکرار بی‌مورد، بی‌نظمی و ورود داده‌های تهی در جداول جلوگیری می‌کند. این کار با نرمال‌سازی جداول اتفاق می‌افتد که در ادامه نرمال‌سازی را می‌بینید.

۳. فیلدهای هر رکورد را در هر جدول مشخص نمایید.

٤. برای هر جدول کلید اولیه و کلیدهای کاندید را تعیین کنید تا تضمین شود که هیچ دو رکورد یکسان نباشند.

۵. ارتباط بین جداول را تعیین کنید تا بتوانید پرس‌وجوها را از چند جدول تهیه کنید.

٦. طراحی را با کاربران مرور کنید تا مطمئن شوید، بانک اطلاعات طراحی شده نیازهای کاربران را برطرف می‌کند.

۷. جداول بانک اطلاعات را ایجاد کرده، داده‌ها را در آنها وارد نمایید.

۸. کارایی بانک اطلاعات را تحلیل کنید و بانک اطلاعات طراحی شده را بهینه نمایید تا بتوانید سریع‌تر نتایج پرس‌وجوها را دریافت کنید.

۳-۱-۲. نرمال‌سازی داده‌ها

نرمال‌سازی، فرآیند تنظیم ساختار جدول‌های بانک اطلاعاتی است. هدف نهایی نرمال‌سازی این است که داده‌های موجود در بانک اطلاعاتی به ساده‌ترین ساختار آن تبدیل شود و داده‌های زاید به حداقل برسند. به عبارت دیگر، اهداف اولیه نرمال‌سازی، کاهش افزونگی داده، کاهش بی‌نظمی و کاهش داده‌های تهی[۱۱] است.

نرمال‌سازی، یک منشأ ریاضی پیچیده دارد که شامل مراحل خاصی به نام **فرم‌های نرمال**[۱۲] است. آقای کاد در مقاله اولیه خود سه فرم نرمال را معرفی کرد که به ۱NF، ۲NF و ۳NF معروف هستند.[۱۳] شخص دیگری به

۱- NULL ۲- Normal Forms ۳- First Normal Form, Second Normal Form, Thrid Normal Form

۴- Boyce- codd Normal Form

نام «بویس» به همراه کاد فرم نرمال دیگری به نام BCNF[۱۴] را تعریف کرد. بعدها دیگران فرم‌های نرمال‌سازی ٤NF و ٥NF را معرفی کردند که نادرند. هرچه فرم نرمال بیشتر باشد، محدودیت تست بیشتر است. تا نرمال‌سازی فرم سوم (٣NF) برای بانک اطلاعاتی کافی است.

☒ **در فرم نرمال‌سازی اول،** تست می‌شود که هیچ فیلدی (ستون) بیش از یک قلم داده را شامل نشود. به عنوان مثال، آدرس باید به بخش‌های کشور، استان، شهرستان، خیابان، کوچه و پلاک تقسیم شود. علاوه بر این، در این فرم نرمال‌سازی داده‌های تکراری حذف می‌شوند. به عنوان مثال، از تکرار بعضی از فیلدهای یک رکورد جلوگیری می‌کند. به زبان ساده‌تر، جدولی فرم نرمال اول است که:

☒ همه کلیدهای آن تعریف شده باشند.

☒ تمام فیلدهای آن به کلید اولیه وابستگی تابعی[۱۵] داشته باشند.

☒ فیلدهای آن دارای دامنه تو در تو نباشند. فیلد **نام** باید به فیلدهای **نام** و **نام‌خانوادگی** تقسیم شوند.

☒ **فرم نرمال دوم،** هرگاه جدولی فرم نرمال اول باشد و فیلدهای آن به بخش‌های کلید اولیه وابسته نباشد، فرم نرمال دوم است.

☒ **فرم نرمال سوم،** هرگاه جدولی فرم نرمال دوم باشد و فیلدهای غیرکلید آن به فیلدهای غیرکلید دیگر وابسته نباشد، فرم نرمال سوم است. به عنوان مثال، با استفاده از فیلدهای **تعداد واحد** و **نمره،** می‌توان معدل دانشجو را محاسبه کرد. در نتیجه نیازی به فیلد معدل در جدول نیست.

توجه داشته باشید که وقتی بانک اطلاعاتی را طراحی می‌نمایید، به نرمال‌سازی جداول آن توجه ویژه‌ای داشته باشید. زیرا، نرمال‌سازی جداول موجب افزایش کارایی خواهد شد.

۲-۲. بانک اطلاعات SQL Server

سیستم بانک اطلاعات SQL Server، یکی از سیستم‌های بانک اطلاعات رابطه‌ای است که قدرت فوق‌العاده‌ای دارد. نسخه‌های مختلفی از SQL Server ارائه شده است و آنچه در این کتاب در مورد SQL Server گفته می‌شود، SQL Server ۲۰۰۸ می‌باشد.

۵- اگر دو فیلد A و B در رابطه R داشته باشیم، آنگاه وابستگی تابعی $A \rightarrow B$ برقرار است اگر برای تمام رابطه‌ها در R، به ازای هر مقدار A فقط یک مقدار B داشته باشیم.

این سیستم مدیریت بانک اطلاعات (DBMS[16])، در حدود ۷۰ درصد از بانک‌های اطلاعات موجـود را بـه خود اختصاص داده است، به طوری که برای سازمان‌های قدیمی و جدید، کوچک و بزرگ قابل استفاده اسـت. به همین دلیل، در این کتاب بانک اطلاعات SQL Server را بررسی می‌کنیم.

SQL Server ویژگی‌های مختلفی دارد که برخی از آنها عبارت‌اند از:

☒ **پشتیبانی از سخت‌افزارهای جدید:** این نسخه علاوه بر CPUهای قـدیمی، پردازشـگرهای جدیـد ٦٤ بیتی را نیز پشتیبانی می‌نماید. بنابراین، حافظه قابل دسـترس و سـرعت ایـن نسـخه بانـک اطلاعـات افـزایش می‌یابد.

☒ **پشتیبانی از انواع جدید:** این نسخه علاوه بر داده‌های معمولی، انواع داده از قبیـل XML و Varbinary را پشتیبانی می‌کند که XML برای ذخیره و بازیابی اسناد XML به کار می‌رود و نوع Varbinary، برای ذخیـره و بازیابی تصویر در فیلدهای جداول به کار می‌روند.

☒ **پشتیبانی از چند نمونه:** بانک اطلاعات SQL Server ۲۰۰۰ حداکثر تا ١٦ نمونه را می‌توانست پشتیبانی کند. ولی، این نسخه از می‌تواند ۵۰ نمونه پشتیبانی نماید.

☒ **پارتیشن‌بندی داده‌ها:** در این نسخه، جداول و شاخص‌های[17] بزرگ را می‌توان به چند پارتیشن تبـدیل نمود و در چندین پارتیشن مختلف قرار داد تا سریع‌تر بتوان به داده‌های خاصی دستیابی پیدا کرد.

☒ **بهبود سرویس‌های گزارش‌گیری:** یکی از ابزارهای بسیار مهم بانک اطلاعات، مدیریت گزارش‌گیـری است. این نسخه ابزار قدرت‌مندی برای تولید گزارش دارد.

☒ **بهبود در کاتالوگ:** کاتالوگ مکانی است که اشیای موجود در بانک اطلاعات را نگهداری مـی‌کنـد. در نسخه SQL Server ۲۰۰۰ و قبل از آن، کاتالوگ به عنوان بخشی از داده‌های اولیه ذخیره می‌شـدند، در حـالی که در این نسخه حدود ۲۵۰ دید[18] جهت نگهداری کاتالوگ در نظر گرفته شده است.

☒ **یک‌پارچگی با فریم‌ورک دات‌نت:** مهـم‌تـرین تحـول نسـخه‌هـای ۲۰۰۵ و ۲۰۰۸ یـک‌پـارچگی بـا فریم‌ورک دات‌نت است. این یک‌پارچگی امکان تولید رویه‌های ذخیره شده[19]، توابعی که توسط کاربر تعریـف می‌شوند، تریگرهـا، کرسـرهـا و غیـره را در یکـی از زبـان‌هـای برنامـه‌نویسـی از قبیـل Vb.NET، C#.NET، VC++.NET و F#.NET می‌دهد.

۱- Database Management System ۲- Indexes ۳- View ۴- Stored Procedures

۳-۲. معرفی بانک اطلاعاتی نمونه

در این بخش یک بانک اطلاعاتی نمونه را معرفی می‌نمایم. سپس، مراحل طراحی و پیاده‌سازی این بانک اطلاعاتی را با هم مرور می‌کنیم. بانک اطلاعاتی که برای این کتاب در نظر گرفته شده است، اطلاعات مربوط به شهرها و استان‌ها را نگهداری می‌نماید. برای طراحی این بانک اطلاعاتی، مراحل زیر باید انجام شود:

۱. **تعیین اهداف بانک اطلاعاتی:** در این مرحله باید فرم‌ها، نمودارها، گزارشات و موارد دیگر مورد نیاز کاربران تعیین شود. در این مرحله، همچنین باید فرم‌ها، گزارشات و غیره طراحی، پیاده‌سازی با کاربران مرور شوند.

۲. **توزیع داده‌ها:** مهم‌ترین مرحله طراحی بانک اطلاعاتی، تعیین چگونگی توزیع داده است. در این مرحله باید تعیین گردد، چه داده‌هایی در چه جداولی توزیع شود. در این بخش باید تعداد جداول بانک اطلاعاتی، فیلدهای هر یک از جداول و نوع فیلدها تعیین شود. در این بانک اطلاعاتی نمونه چند جدول در نظر گرفته شده است که برخی از آنها عبارت‌اند از:

☒ **جدول City:** اطلاعات مربوط به شهرها از قبیل کد شهر، نام شهر و کد استان مربوط به آن شهر را نگهداری می‌کند.

☒ **جدول State:** اطلاعات مربوط به استان‌ها از قبیل کد استان و نام استان را نگهداری می‌کند (جدول۱-۲ را ببینید).

☒ **جدول User:** اطلاعات کاربران بانک اطلاعات را نگهداری می‌کند. این اطلاعات عبارت‌اند از: کد کاربر، نام کاربر، کلمه عبور، نام‌خانوادگی کاربر، سطح دسترسی و تصویر کاربر (جدول۱-۲ را مشاهده کنید).

۳. **تعیین ساختار و فیلدهای جدول:** بعد از این که فیلدهای جدول مشخص گردید، باید تعیین شود نوع هر فیلد چیست، آیا فیلد می‌توانند Null باشد یا خیر، چه فضایی از حافظه را اشغال می‌کنند. در این بخش باید تعیین کنیم هر جدول از چند فیلد تشکیل شده است و هر فیلد دارای چه خواصی (نام، اندازه، نوع و محدودیت‌ها) است. فیلدهای بانک اطلاعاتی نمونه در جدول۱-۲ آمده‌اند.

٤. **تعیین فیلد کلید اولیه[۲۰] جداول:** یکی از بخش‌های مهم طراحی بانک اطلاعاتی تعیین فیلد کلید اولیه هر جدول است. فیلد کلید اولیه سه خاصیت دارد که عبارت‌اند از:

۱. فیلد کلید اولیه نمی‌تواند تهی (Null) باشد. یعنی، اگر فیلدی را به عنوان کلید اولیه انتخاب کنید، به طور خودکار محدودیت NOT NULL به آن تخصیص می‌یابد.

---

[۲۰]- Primary Key [۲]- Views [۳]- Functions [۴]- Stored Procedures [۵]- Cursors [۶]- Triggers

۲. فیلد کلید اولیه (یا کلید) تضمین می‌کند که هیچ دو رکوردی در جدول برای آن فیلد دارای مقدار یکسانی نمی‌باشند. یعنی، فیلد کلید از ورود رکوردهای تکراری در جدول جلوگیری می‌کند.

۳. اطلاعات جدول براساس فیلد کلید اولیه مرتب می‌شوند. به عنوان مثال، در بانک اطلاعاتی نمونه، در جدول User، فیلد Account، فیلد کلید اولیه است. زیرا، برای هیچ دو کاربری مقدار این فیلد برابر نمی‌باشد، یا به عبارت دیگر، هیچ دو کاربری حساب کاربری یکسانی ندارند. ولی، در جدول City، فیلد CityCode فیلد کلید اولیه است.

۵. تعیین ارتباط بین جدول‌ها: یکی از ویژگی‌های اولیه بانک اطلاعات رابطه‌ای، ارتباطی است که جداول می‌توانند با یکدیگر داشته باشند. ارتباط بین جداول موجب می‌شود تا از افزونگی (تکرار بی‌مورد داده‌ها در جدول) جلوگیری شود. ارتباط بین جدول City و State با استفاده از فیلد StateCode برقرار می‌شود.

٦. تعیین اشیای دیگر بانک اطلاعات: بانک اطلاعاتی از اشیای متعددی تشکیل شده است. مهم‌ترین شیء بانک اطلاعاتی جدول است. جدول برای نگهداری داده‌های بانک اطلاعاتی به کار می‌رود. اشیای دیگر از قبیل دیدها[۲۱]، توابع[۲۲]، رویه‌های ذخیره شده[۲۳]، کرسرها[۲۴]، تریگرها[۲۵] و غیره در بانک اطلاعاتی وجود دارند.

در ادامه کتاب با این اشیاء آشنا خواهید شد و چگونگی ایجاد، حذف و ویرایش آن‌ها را در بانک اطلاعاتی می‌آموزیم.

فصل ۳ مفاهیم اولیه و عملگرهای LINQ

کد و داده دو عنصر اساسی در نرم افزار هستند که هر یک جایگاه ویژه‌ای دارند. یکی از بخش‌های بسیار مهم نرم‌افزار در زمان پیاده‌سازی، نوشتن کدهایی برای دستیابی به داده است. پیاده‌سازان می‌توانند از زبان‌های متعددی برای دستیابی به داده استفاده کنند. انتخاب زبان برنامه نویسی به عوامل مختلفی از قبیل نوع رفتار برنامه، پیش‌زمینه و دانش پیاده‌سازان، مهارت‌های تیم پیاده‌سازی، نوع سیستم‌عامل، سیاست‌های سازمانی و عوامل دیگر بستگی دارد. از طرف دیگر، داده‌ها بر روی دیسک به صورت فایل متنی یا غیره، جداولی در بانک اطلاعات و یا اسناد XML ذخیره می‌شوند که باید توسط برنامه بازیابی، پردازش و به یکدیگر تبدیل گردند.

بنابراین برنامه نویسان برای پیاده‌سازی برنامه حداقل به سه ابزار بسیار مهم نیاز دارند که عبارت‌اند از:

۱.یکی از زبان‌های برنامه نویسی شی‌گرا نظیر زبان‌های برنامه نویسی دات‌نت یا جاوا.

۲.ابزاری برای مدیریت بر بانک اطلاعات

۳.ابزاری برای مدیریت بر اسناد XML

یکی از مهم‌ترین مشکلاتی که برنامه نویسان با آن مواجه هستند این است که هر یک از این ابزارها از مدل داده‌ای[۲۶] متفاوتی پیروی می‌کنند. مدل داده‌ای در بانک اطلاعات معمولا زبان SQL[۲۷] است. مدل داده‌های اسناد XML نیز به گونه دیگری است. در XML باید از زبان XQurey استفاده کنید. در حالی که هر زبان برنامه‌نویسی نیز از مدل داده‌ای خاصی استفاده می‌کند. به عنوان مثال، دات‌نت از فناوری ADO.NET برای دستیابی به داده‌ها استفاده می‌کند. بنابراین، برنامه‌نویس باید داده‌های خود را از مدلی به مدل دیگر تبدیل کند تا بتواند از داده‌ها استفاده کند. به عنوان مثال، یک برنامه مدیریت فروش اینترنتی را در نظر بگیرید. در این برنامه، گاهی برنامه‌نویس نیاز دارد اطلاعات (نظیر اوراکل، MySQL، SQL Server، DB۲ و اکسس) را خوانده به فرمت XML تبدیل کند تا بتواند از طریق اینترنت ارسال نماید. در این صورت، برنامه‌نویس مجبور است داده‌ها را

---

۱. Data Model ۲ . Structure Query Language ۳ .Bussiness Teir

ابتدا از مدل بانک اطلاعات با زبان SQL بخواند و سپس به لایه کسب و کار[۲۸] مانند ADO.NET تبـدیل کـرده، آن را پردازش نماید و در نهایت با زبان XQuery آن را به فرمت اسناد XML تبدیل کـرده، تـا بتوانـد از طریـق اینترنت به برنامه‌های دیگر ارسال کند.

این تبدیلات نه تنها زمان بر است، بلکه برنامه‌نویس باید کار کردن با ابزارهای مختلفی را یاد بگیرد تا بتواند از این ابزارها استفاده کند. به همین دلیل، شرکت مایکروسافت برای حل این مشکلات، بررسی دو راهکار را در دستور کار خود قرار داد:

۱. ایجاد امکانات مورد نیاز متخصصین XML یا داده‌های رابطه‌ای (بانک اطلاعات) در هر یک از زبـان‌هـای برنامه‌نویسی در زمان اجرا. این پیشنهاد نه تنها مشکل اصلی (تبدیلات داده‌ها از مدلی به مـدل دیگـر) را حـل نمی‌کرد، بلکه در مواردی مانند نگهداری، مشکلات را افزایش می‌داد.

۲. اضافه کردن قابلیت‌هـایی بـرای نوشـتن پـرس‌وجوهـای همـه منظـوره در فـریم‌ورک دات‌نـت. شـرکت مایکروسافت این راهکار را پذیرفت و در سال ۲۰۰۳ شروع به پیاده سازی تکنولوژی جدیدی به نام LINQ [۲۹] (لینک) کرد تا بتواند بر این مشکلات غلبه کند. این تکنولوژی (LINQ) در ۱۹ نـوامبر سـال ۲۰۰۷ بـه عنـوان قسمتی از فریم‌ورک دات‌نت ۳.۵ عرضه گردید. از آنجایی که منطق و دسـتورات لینـک شـبیه بـه زبـان SQL است، برنامه‌نویسان به سادگی می‌توانند از آن استفاده کنند.

۱-۳. مزایای LINQ

استفاده از LINQ مزایای متعددی دارد که برخی از آن‌ها عبارت‌اند از:

۱. LINQ نوشتن پرس‌وجوها را ساده‌تر می‌نماید. به ویژه زمانی که با یکی از زبان‌های NET. کـار مـی‌کنیـد، نیازی به یاد گیری مطالب زیادی نیست تا پرس‌وجوهای LINQ را بنویسید.

۲. یکی دیگر از ویژگی‌های بسیار مهم این ابزار این است که پرس‌وجو برای منابع داده‌ای مختلف را یکسـان کرده است. یعنی، می‌توانید به همان روشی که بر روی بانک اطلاعات SQL Server پرس‌وجو تهیه مـی‌کنیـد، روی Dataset، یا اسناد XML و حتی کلکسیون‌های دات‌نت نیز پرس‌وجو تهیه کنید.

۳. لینک بین داده‌های رابطه‌ای[۳۰] و دنیای شی‌گرایی ارتباط ایجاد کرده است. یعنی، با لینک به سادگی می‌توان ارتباط بین اشیاء و داده‌ها را تعریف کرد.

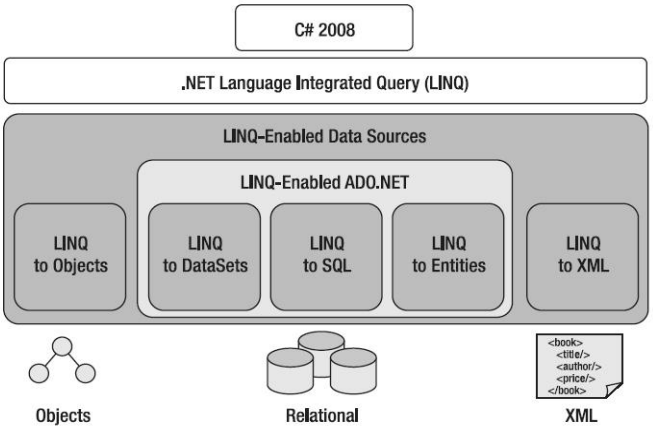
۱.Language Integrated Query ۲.Relational Data ۳.Application Programming Interface

٤. لینک سرعت تولید نرم‌افزار را افزایش می‌دهد. زیرا، خطاهای زمان کامپایل مربوط به پرس‌وجوها را می‌توان با ابزار Debug ویژوال استودیودات‌نت تشخیص داده و آن‌ها را تصحیح نمود.

۲-۳. معماری LINQ

LINQ، تکنولوژی جدیدی است که داده‌های زیادی را تحت پوشش قرار می‌دهد. لینک، علاوه بر این که داده‌های بانک اطلاعات را پشتیبانی می‌کند، داده‌های شی‌گرا و اسناد XML را نیز پشتیبانی می‌نماید. معماری لینک در شکل ۱-۳ آمده است. همان طور که در این شکل می‌بینید، لینک از قطعات زیر تشکیل شده است:

۱. قطعه LINQ to Objects، یک واسط برنامه‌نویسی کاربردی (API[۳۱]) است که عملگرهایی برای ایجاد پرس‌وجوهای استاندارد دارد. این قطعه، قابلیت‌هایی دارد که اجرای پرس‌وجو برای اشیای موجود در حافظه از قبیل آرایه‌ها، کلکسیون‌ها و دیگر اشیا را فراهم می‌کند. این قابلیت بر روی تمامی کلاس‌هایی که کلکسیون اینترفیس Inumerable را پیاده‌سازی کرده است، قابل اجرا می‌باشد.



شکل ۱-۳ معماری تکنولوژی LINQ.

۲. قطعه LINQ to XML، قابیلت اجرای پرس‌وجوها روی اسناد XML را فراهم می‌کند. یعنی، عملگرهایی را تعریف می‌کند که برای تولید پرس‌وجو بر روی اسناد XML به کار می‌روند. با این قطعه می‌توانید اعمالی از قبیل بازیابی داده از اسناد XML، ایجاد اسناد XML، بارکردن اسناد XML، دستکاری و ویرایش اسناد XML را

انجام دهید. این قطعه، عملگرهایی دارد که به XPath وابسته است. این عملگرها امکان حرکت در بین گره‌های مختلف اسناد XML را می‌دهند.

۳. قطعه LINQ to ADO.NET، اجرای پرس‌وجوهای LINQ بـر روی بانـک اطلاعـات رابطـه‌ای را فـراهم می‌کند. این قطعه از سه قطعه زیر تشکیل شده است:

❖ **قطعه LINQ to SQL**، بخشی از ADO.NET است که برای تولید پرس‌وجو بر روی بانک‌های اطلاعات رابطه‌ای به کار می‌رود. این قطعه، برای نگاشت[۳۲] مدل داده مربوط به شـیء بانـک اطلاعـات رابطـه‌ای از قبیـل جدول به یک مدل شیء تعریف شده توسط برنامه‌نویس در زبان برنامه‌نویسی به کار می‌رود.

❖ **قطعه LINQ to Dataset**، بخشی از ADO.NET است که برای تولید پرس‌وجو بر روی Dataset به کـار می‌رود.

❖ **قطعـه LINQ to Entities**، بخشـی از ADO.NET اسـت کـه بـه برنامـه‌نـویس اجـازه مـی‌دهـد موجودیت‌ها[۳۳] را به صورت کلاس‌هایی تعریف کرده آن‌ها را در برنامه استفاده کند. به طـوری کـه بـه سـادگی بتوان این موجودیت‌ها را در بانک اطلاعات ذخیره نمود.

۳ – ۳. اسمبلی‌های میانی هسته LINQ

هسته LINQ از یک سری اسمبلی‌هایی تشکیل شده است که این اسمبلی‌ها و وظـایف آن‌هـا را در جـدول ۱-۳ می‌بینید.

جدول ۱-۳ اسمبلی‌های تشکیل دهنده LINQ و وظایف آن‌ها.

نام اسمبلی	هدف
System.Query.DLL	انواعی که هسته API مربوط به LINQ را نمایش می‌دهند، تعریف می‌کند.
System.Data.Dlinq.DLL	امکانات مورد نیاز برای استفاده LINQ با بانک اطلاعات رابطه‌ای (LINQ to SQL) را فراهم می‌کند.
System.Data.Extension.DLL	تعداد متد اضافی برای کار با ADO.NET را فراهم می‌نماید.
System.Xml.Xlinq.DLL	امکاناتی را جهت کار اسناد XML در LINQ فـراهم مـی‌کنـد (LINQ to

۱.Mapping ۲.Entities

	(XML

فضای نام مورد نظر برای کار با LINQ

برای استفاده از LINQ باید فضای نام System.Linq به پروژه اضافه گردد. البته در ویژوال استودیو دات‌نت ۲۰۰۸ و نخسه‌های بالاتر از آن، وقتی یک پروژه جدیدی ایجاد می‌کنید، این فضای نام به طور خودکار به پروژه اضافه می‌گردد. اگر این فضای نام به پروژه اضافه نگردید، با دستور using می‌توانید آن را به پروژه اضافه نمایید (دستور زیر):

```
using System.Linq;
```

۳-۴. LINQ چطور پیاده‌سازی می‌شود؟

LINQ، یک بسط و توسعه دات‌نت است. گرامر آن جدید می‌باشد و عناصر جدیدی به زبان‌هایی از قبیل C# و VB.NET اضافه کرده است. پرس‌وجوهای LINQ، به عنوان فراخوانی‌های عمومی (کلی) منتشر می‌شوند و بخش‌های جدایی از فریم‌ورک دات‌نت هستند.

۳-۵. به کارگیری عملگرهای پرس‌وجوی استاندارد

کلمات کلیدی زیادی وجود دارند که گرامر LINQ را تولید می‌کنند. بعضی از این عملگرها شبیه زبان پرس‌وجوی ساخت یافته (SQL) هستند و برخی دیگر متفاوت‌اند. در ادامه با این عملگرها آشنا خواهید شد.

۳-۶. تعریف متغیر برای نگهداری نتیجه پرس‌وجوی LINQ

برای این که بتوانید نتیجه پرس‌وجوهای LINQ را نگهداری کنید، باید متغیری را تعریف کرده تا نتیجه پرس‌وجوی LINQ را در آن قرار دهید. برای این منظور، می‌توانید از کلمه کلیدی var به صورت زیر استفاده کنید:

```
var    نام متغیر  =  LINQ  پرس‌وجو ;
```

var، کلمه کلیدی است که برای تعریف نوع‌های کلی به کار می‌رود. یعنی، این نوع به کامپایلر می‌گوید که با اولین بار مقداردهی به این متغیر، نوع آن مشخص می‌شود. به عنوان مثال، دستورات زیر را ببینید:

```
var  x = ۱۰ ;
```

```
var  Fname = "Ali";
```

این دستورات، دو متغیر به نام‌های x و Fname از نوع var تعریف می‌کنند که x را برابر بـا ۱۰ و Fname را برابر با "Ali" قرار می‌دهند. در ادامه خواهید دید به این نوع متغیرها می‌توان پـرس‌وجوهـای LINQ را نسـبت داد.

۳-۷ . ایجاد پرس‌وجو از منابع داده

به دو روش می‌توان پرس‌وجوها را بر روی منابع داده انجام داد. این دو روش عبارت‌اند از:

۱.استفاده از ساختار پرس‌وجوی LINQ

۲.استفاده از متدهای پرس‌وجو و عبارات lambda

۱-۷-۳ . ساختار پرس‌وجوی LINQ

اولین مرحله استفاده از پرس‌وجوی LINQ، شناخت ساختار آن است. این ساختار در زیر آمده است:

```
from <element> in <data_source>
[query standard operators]
select <element type in returned sequence>
```

بخش‌های این ساختار در زیر آمده است:

❖ **کلمه کلیدی from**، تمامی پرس‌وجوهای LINQ با این کلمه شروع می‌شوند. یعنـی، ایـن کلمـه نشـان دهنده شروع پرس‌وجوی LINQ است.

❖ **بخش element**، یکی از عناصر منبع داده (data_source) را تعیین می‌کند که می‌خواهیم از آن پرس‌وجو تهیه کنیم.

❖ **کلمه کلیدی in**، بخش سوم پـرس‌وجـو LINQ ایـن کلمـه کلیـدی اسـت کـه در هنگـام ایجـاد کلیـه پرس‌وجوها اجباری می‌باشد.

❖ **بخش منبع داده (data _source)**، منبع داده‌ای را تعیین می‌کند که پرس‌وجو باید بر روی آن اجرا شود. منبع داده می‌تواند یک آرایه، لیست، سند XML، کلکسیون یا جدول بانک اطلاعاتی باشد.

❖ **بخش عملگرهای استاندارد پرس‌وجـو (query standard operators)**، تمـام عملگرهـای اسـتاندارد LINQ هستند که برای اعمالی از قبیل فیلتر کردن، مرتب‌سازی، گروه بندی، ادغام و اعمال دیگر به کار می‌روند. مهم‌ترین این عملگرها where است که برای فیلتر کردن اطلاعات منبع داده به کار می‌رود. عبـارت شـرطی کـه

برای فیلتر کردن داده استفاده می‌شود، در جلوی کلمه کلیدی where قرار می‌گیرد. به عنوان مثال، دستور زیر را ببینید:

```
where  n.StartWith ("S") ;
```

این دستور، عناصری را برمی‌گرداند که با "s" شروع می‌شوند.

❖ **بخش select برای انتخاب عناصر پرس‌وجو**، عناصری را تعیین می‌کند که باید در نتیجه پرس‌وجو آورده شوند. هر پرس‌وجوی LINQ با کلمه کلیدی select خاتمه می‌یابد.

❖ **بخش نوع عنصر در ترتیب برگردانده شده(Element type in returned sequence)**، نوع عناصر موجود در لیست ایجاد شده توسط پرس‌وجو را تعیین می‌کند. در این بخش می‌توان نوع خروجی‌های پرس‌وجو را به دلخواه تعیین کرد. به عنوان مثال، دستورات زیر را ببینید:

```
int[] nums = { ٤, ١, ٧, ٥, ٩, ٦, ٢ };
var query = from n in nums  where n > ٤    select n.ToString();
foreach (string s in query)
  Console.Write(s);
```

دستور اول، آرایه‌ای به نام nums تعریف می‌کند و مقدار اولیه به عناصر آن تخصیص می‌دهد. دستور دوم، همان طور که بیان گردید، پرس‌وجو را با کلمه کلیدی from شروع می‌کند. n در این پرس‌وجو نشان دهنده یکی از عناصر موجود در منبع داده است که از آن برای فیلتر کردن استفاده می‌کند. بعد از n کلمه کلیدی in و بعد از آن منبع داده (آرایه nums) قرار گرفته است. برای فیلتر کردن از دستور where استفاده کرد و در بخش select اطلاعاتی که باید در پرس‌وجو آورده شوند، را مشخص کرده است. یعنی، این پرس‌وجو، مقادیری که بزرگتر از ٤ هستند را به رشته تبدیل می‌کند و برمی‌گرداند. دستور سوم، با foreach، رشته را نمایش می‌دهد.

۲-۷-۳ . استفاده از متدهای پرس‌وجو و عبارات Lambda

یکی از روش‌های ایجاد پرس‌وجو استفاده از متدهای پرس‌وجو و عبارات Lambda است. به عنوان مثال، دستورات زیر را مشاهده کنید:

```
int[] nums = { ٤, ١, ٧, ٥, ٩, ٦, ٢ };
var query = nums.Where(n => n > ٤).Select(n => n.ToString());
foreach (string s in query)
    Console.Write(s);
```

این دستورات، اعدادی که بزرگتر از ٤ باشند، را به رشته تبدیل می‌کنند و نمایش می‌دهند. در ادامه، با این متدها و عبارات Lambda بیشتر آشنا خواهید شد.

اکنون دستورات زیر را ببینید:

```
int[] nums = { ٤, ١, ٧, ٥, ٩, ٦, ٢ };
var query=nums.Where((n,index)=>n>٤ && index٪٢==٠).Select(n=> n.ToString());
foreach (string s in query)
  Console.Write(s);
```

دستور اول، آرایه‌ای را تعریف کرده مقادیری را به آن تخصیص می‌دهد. دستور دوم، مقادیری از آرایه که اندیس آن‌ها زوج باشد را برمی‌گرداند. و دستور سوم، با حلقه foreach آن‌ها را نمایش می‌دهد.

۸–۳ . عملگرهای استاندارد پرس‌وجو

LINQ مانند SQL از یک‌سری عملگرهای استاندارد تشکیل شده است. این عملگرها، برای ایجاد پرس‌وجوها در LINQ به کار می‌روند. عملگرهای استاندارد پرس‌وجو[۳٤] در فضای نام System.Linq قرار دارند. هرگاه پروژه جدیدی اضافه کنید، این فضای نام به طور خودکار به پروژه اضافه خواهد شد. این فضای نام، عملگرهای زیادی را پشتیبانی می‌کند که برخی از آن‌ها در جدول ۲ – ۳ آمده‌اند.

برای این که بتوانیم عملگرهای LINQ را شرح دهیم، ابتدا کلاس‌هایی تعریف می‌کنیم تا عملگرها را بر روی این کلاس‌ها آزمایش نماییم. در این بخش به دو کلاس زیر می‌پردازیم:

۱. **کلاس State**، اطلاعات استان‌ها را نگهداری می‌کند و به صورت زیر تعریف می‌گردد:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace test
{
    public class State
    {
        private string _stateCode;
        private string _stateName;
        public string stateCode
        {
            get{ return _stateCode; }
            set { _stateCode = value;  }
        }

        public string stateName
```

۱.Standard Query Operators

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

این کلاس از اعضای زیر تشکیل شده است:

- فیلدهای stateCode_ (کد استان را نگهداری می‌کند) وstateName_ (نام استان را نگهداری می‌نماید).
- خواص StateCode (برای دستیابی و مقداردهی فیلد stateCode_ به کار می‌رود) و StateName (برای دستیابی و مقداردهی فیلد stateName_ به کار می‌رود).

۲. کلاس City، اطلاعات شهرها و کد استان آن‌ها را نگهداری می‌کند و به صورت زیر تعریف می‌شود:

```
Using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace test
{
    public class City
    {
        private string _cityCode;
        private string _cityName;
        private string _stateCode;
        public string cityCode
        {
            get {  return _cityCode;     }
            set {  _cityCode = value;    }
        }
        public string cityName
        {
            get  {  return _cityName;    }
            set  {  _cityName = value;   }
        }
          public string stateCode
          {
            get  { return _stateCode;    }
            set  { _stateCode = value;   }
        }
    }
}
```

این کلاس دارای اعضای زیر می‌باشد:

❖ فیلدهای cityCode_ (کد شهرستان را نگهداری می‌کند)، cityName_ (نام شهرستان را نگهداری می‌نماید) وstateCode_ (کد استان مربوط به شهرستان را نگهداری می‌نماید).

❖ خواص CityCode (برای دستیابی و مقداردهی فیلد cityCode_)، cityName (برای دستیابی و مقداردهی فیلد cityName_) و stateCode (برای دستیابی و مقداردهی فیلد stateCode_).

اکنون دستورات زیر را ببینید:

```
List<City> city = new List<City>
{
  new City{cityCode ="۰۱", cityName = "بابل", stateCode ="۰۱"},
  new City{cityCode ="۰۲", cityName ="آمل", stateCode="۰۱"},
  new City{cityCode="۰۳", cityName="ساري", stateCode="۰۱"},
  new City{cityCode="۰٤", cityName= "چالوس",stateCode= "۰۱"},
  new City{cityCode="۰٥", cityName="گرگان",stateCode="۰۲"},
  new City{cityCode = "۰٦", cityName = "گنبد",stateCode ="۰۲"},
  new City{cityCode="۰۷", cityName="بجنورد",stateCode="۰۳"},
  new City{cityCode="۰۸", cityName="قوچان",stateCode="۰۳"}
};
List<State> state = new List<State>
{
  new State {stateCode = "۰۱", stateName  = "مازندران"},
  new State {stateCode = "۰۲", stateName  = "گلستان"},
  new State {stateCode = "۰۳", stateName  = "خراسان جنوبي"},
  new State {stateCode = "۰٤", stateName  = "خراسان رضوي"}
 };
```

این دستورات دو کلکسیون از نوع‌های City (به نام city) و State (به نام state) ایجاد کرده، لیست برخی شهرستان‌ها و استان‌ها را به آن‌ها اضافه می‌کنند.

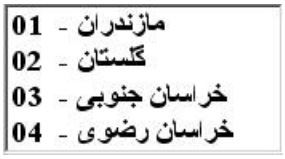
در ادامه این فصل، مثال‌هایی از عملگرها آورده شده‌اند که خروجی آن‌ها با توجه به این داده‌های اضافه شده می‌باشند.

عملگر Select

این عملگر برای بازیابی مقادیر از کلکسیون‌ها، اسناد xml، آرایه‌ها، لیست‌ها و جداول بانک اطلاعات به کار می‌رود (مانند دستور SELECT در SQL). به عنوان مثال، دستورات زیر را ببینید:

```
var query = from o in State    select o;
foreach (var i in query)
  listBox١.Items.Add(i.stateCode + "  - " + i.stateName)
```

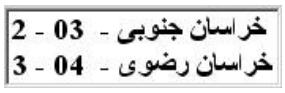
دستور اول، اطلاعات مربوط به استان‌ها را در query ذخیره می‌کند و دستور foreach اطلاعات query (هر یک از عناصر آن) را بر روی listBox۱ نمایش می‌دهد (شکل مقابل).



اکنون دستورات زیر را مشاهد کنید:

```
var query۱ = State.Select((o, index) => new { Position = index, o.stateCode,
     o.stateName }).Where(o => o.stateName.StartsWith("خ"));
foreach (var i in query۱)
  listBox۲.Items.Add(i.Position.ToString()+"-"+i.stateCode+"-"+i.stateName);
```

دستور اول، شماره مکان، کد استان و نام استان‌هایی را بازیابی می‌کند که نام استان آن‌ها با حرف "خ" شروع شده باشد و دستور foreach،



این اطلاعات را به listBox۲ اضافه می‌کنند (شکل مقابل).

در این پرس‌وجو از عملگر where استفاده گردید که در ادامه عملکرد آن را خواهید دید.

عملگر Where

این عملگر، عناصری از کلکسیونی را برمی‌گرداند که دارای شرط خاصی باشند. به عنوان مثال، دستورات زیر را ببینید:

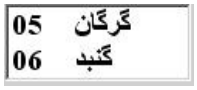
```
var query = city.Where((c, index) => index >۵ );
foreach (var  item in query)
   listBox۱.Items.Add(item.cityCode + "    "+ item.cityName );
```

دستور اول، عناصری از کلکسیون city را بازیابی می‌کند که مکان آن بیشتر از ۵ باشد و حلقه تکرار for، مقادیر cityCode (کدشهر) و cityName (نام شهر) آن‌ها را نمایش می‌دهد (خروجی مقابل).

اکنون دستورات زیر را ببینید:

```
var query = from c in city  where c.stateCode == "۰۲"     select c;
foreach (var  item in query)
  listBox۱.Items.Add(item.cityCode + "    "+ item.cityName );
```

دستور اول، عناصری از کلکسیون شهر (city) را برمی‌گرداند که کد استان آن‌ها برابر با ۰۲ باشد و حلقه تکرار کد شهر و نام شهر را نمایش می‌دهد(شکل مقابل).



عملگر SelectMany

این عملگر مانند عملگر Select عمل می‌کند. با این تفاوت که می‌توان چندین بخش from را با یکدیگر ترکیب کرده و نتیجه را با هم ادغام نماید و به عنوان یک کلکسیون برگرداند. به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
var query = from o in State
    where o.stateCode == textBox۱.Text
    from c in city
     where c.stateCode == o.stateCode
     select new { o.stateName, c.cityName };
foreach (var i in query)
  listBox۱.Items.Add(i.cityName + "  - " + i.stateName);
```

دستور اول، دو بخش from (State و City) را با هم ترکیب کرده و فیلدهای نام استان و نام شهر مربوط به رکوردهایی را بازیابی می‌کند که کد استان آن‌ها در textBox۱ وارد شده باشد. به عنوان مثال، اگر در textBox۱ مقدار ۰۱ وارد شود، خروجی مقابل در listBox۱ نمایش داده می‌شود.

اکنون دستورات زیر را ببینید:

```
 var query۱ = city.Where(c => c.cityCode == textBox۱.Text).SelectMany(c =>
    state.Where(o => o.stateCode == c.stateCode).Select(o => new {
    c.cityName, o.stateName }));
foreach (var i in query۱)
  listBox۲.Items.Add(i.cityName + "  - " + i.stateName);
```

در دستور اول، بخش اول، شهرستانی را برمی‌گرداند که کد شهرستان آن در textBox۱ وارد شده باشد و بخش SelectMany نام شهرستان و نام استانی را برمی‌گرداند که کد شهرستان برابر کد استان (o.stateCode = = c.cityCode) باشد. و دستور دوم، با حلقه foreach، نتیجه پرس‌وجو را نمایش می‌دهد. به عنوان مثال، اگر در textBox۱ مقدار ۰۳ وارد شود، خروجی زیر نمایش داده می‌شود:

ساری - مازندران

عملگر OrderBy

این عملگر، نتایج پرس‌وجو را مرتب می‌نماید. مرتب کردن می‌تواند صعودی یا نزولی باشد. اگر نوع مرتب‌سازی ذکر نشود، مقادیر به صورت صعودی مرتب خواهند شد. ولی اگر بخواهید مقادیر را به صورت نزولی مرتب کنید، باید از گزینه descending استفاده نمایید. چنانچه بخواهید نتایج را براساس چند مقدار مرتب کنید، مقادیر را با کاما از یکدیگر جدا نمایید. به عنوان مثال، دستورات زیر را ببینید:

```
if (checkBox۱.Checked == true)
{
  var query = from c in city
     orderby c.stateCode, c.cityName descending
     select new { c.stateCode, c.cityName };
  foreach (var i in query)
    listBox۱.Items.Add(i.cityName + "  - " + i.stateCode);
}
else
 {
   var query = from c in city
       orderby c.stateCode, c.cityName
       select new { c.stateCode, c.cityName };
  foreach (var i in query)
    listBox۱.Items.Add(i.cityName + "  - " + i.stateCode);
 }
```

دستور اول، بررسی می‌کند checkBox۱ انتخاب شده است یا خیر (اگر checkBox۱ انتخاب شده باشد، اطلاعات را به صورت نزولی مرتب می‌کند). دستور داخل بلاک if، اطلاعات را براساس کد استان به صورت صعودی و نام شهر به صورت نزولی مرتب کرده در query قرار می‌دهد و با دستور foreach اطلاعات مرتب شده را نمایش می‌دهد. ولی، دستورات بلاک else، اطلاعات را براساس کد استان و نام شهر به صورت صعودی مرتب می‌نمایند و دستور foreach این اطلاعات را نمایش می‌دهد. اگر کاربر در برنامه کنید، checkBox۱ انتخاب شده باشد، خروجی مقابل نمایش داده می‌شود.

اکنون دستورات زیر را ببینید:

```
var query = city.Select(c => new {c.cityCode , c.cityName }).OrderBy(c =>
    c.cityName );
foreach (var i in query)
  listBox\.Items.Add(i.cityCode + "  - " + i.cityName);
```

دستور اول، اطلاعات شهرها را براساس نام شهر مرتب کرده فیلدهای کد شهر و نام شهر را در query قرار می‌دهد و دستور foreach این اطلاعات را نمایش می‌دهد (شکل مقابل).

مثال ۱ – ۳ برنامه‌ای که استفاده از عملگرهایی از قبیل Select، SelectMany، Join، GroupBy، TakeWhile و غیره را بیان می‌کند.

مراحل طراحی و اجرا

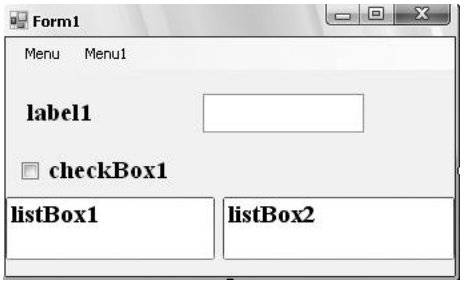
۱. پروژه جدیدی به نام ۱ - ۳ ایجاد کنید.

۲. یک کنترل Label، یک کنترل TextBox، یک کنترل CheckBox و دو کنترل ListBox به فرم برنامه اضافه کنید.

۳. یک کنترل MenuStrip به فرم برنامه اضافه نمایید. گزینه Type Here را کلیک کرده، گزینه Menu را تایپ نمایید.

٤. مکان‌نما را به زیر منوی Menu برده گزینه‌های Where، Select، SelectMany، Join، groupBy، takeWhile، orderBy، ThenBy، Calculate، Distinct، Union، Intersect، Except، Repeat، Range، Reverse، First_last_or_default، Single_or_default، DefaultIfEmpty، ElementAT و ElementAtOrDefault را اضافه کنید.

۵. منوی Menu۱ را جلوی منوی Menu اضافه کنید و گزینه‌های SequenceEqual، All، Any، Contains، Skip، SkipWhile، toLookUP، toList، toDictionary، toArray، ofType و Cast را زیر آن اضافه نمایید.



۶. کلاس جدیدی به نام City به پروژه اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace _۳_۱
{
  public class City
  {
    private string _cityCode;
    private string _cityName;
    private string _stateCode;
    public string cityCode
    {
      get {  return _cityCode;     }
      set {  _cityCode = value;    }
    }
    public string cityName
     {
      get  {  return _cityName;    }
      set  {  _cityName = value;   }
     }
      public string stateCode
     {
       get  { return _stateCode;    }
       set  { _stateCode = value;   }
     }
  }
}
```

این دستورات کلاس شهر (City) را ایجاد می‌کنند که قبلاً عملکرد آن را دیدید.

۷. کلاس جدید دیگری به نام State اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace _۳_۱
{
  public class State
   {
     private string _stateCode;
     private string _stateName;
     public string stateCode
    {
      get{ return _stateCode; }
      set { _stateCode = value;  }
```

```
    }
   public string stateName
    {
     get { return _stateName;   }
     set { _stateName = value;  }
    }
   }
}
```

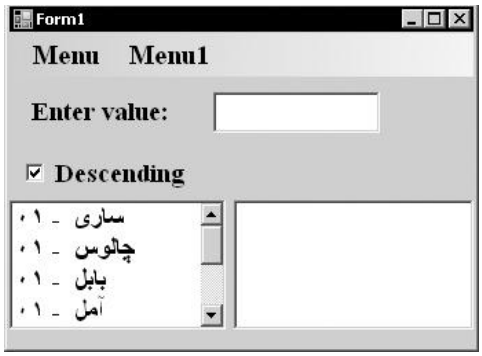
این دستورات، کلاس استان (State) را تعریف می‌کنند.

۸. ناحیه خالی فرم را کلیک مضاعف کرده، به قبل از رویداد Form۱_Load بروید و دستورات زیر را تایـپ کنید:

```
List<City> city = new List<City>
{
   new City{cityCode ="۰۱", cityName = "بابل", stateCode ="۰۱"},
   new City{cityCode ="۰۲", cityName ="آمل", stateCode="۰۱"},
   new City{cityCode="۰۳", cityName="ساری", stateCode="۰۱"},
   new City{cityCode="۰٤", cityName= "چالوس",stateCode= "۰۱"},
   new City{cityCode="۰٥", cityName="گرگان",stateCode="۰۲"},
   new City{cityCode= "۰٦", cityName = "گنبد",stateCode ="۰۲"},
   new City{cityCode="۰۷", cityName="بجنورد",stateCode="۰۳"},
   new City{cityCode="۰۸", cityName="قوچان",stateCode="۰۳"}
};
List<State> state = new List<State>
{
   new State {stateCode = "۰۱", stateName  = "مازندران"},
   new State {stateCode = "۰۲", stateName  = "گلستان"},
   new State {stateCode = "۰۳", stateName  = "خراسان جنوبی"},
   new State {stateCode = "۰٤", stateName  = "خراسان رضوی"}
};
```

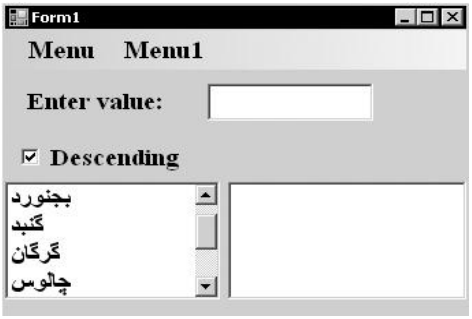
این دستورات، لیستی از شهرها و استان‌ها را تعریف کرده، اطلاعات برخی از شهر و اسـتان‌هـا را در آن‌هـا قرار می‌دهند.

۲۰. پروژه را ذخیره و اجرا کنید تا خروجی برنامه ظاهر شود. گزینه Descending را انتخاب کرده در منوی Menu گزینه orderBy را کلیک نمایید تا شکل ۳ – ۳ ظاهر شود.



شکل ۳ – ۳ مرتب سازی بر اساس نام و کد استان و شهرستان به صورت نزولی.

اکنون گزینه Reverse را کلیک کنید تا شکل ٤ – ۳ ظاهر گردد. گزینه‌های دیگر را امتحان کنید.



شکل ۴ – ۳ نمایش اطلاعات شهرها به صورت معکوس.

فصل ۴ مدل شیء LINQ to SQL

مدل شیء LINQ to SQL، عناصر اصلی برای کارکردن و مدیریت اشیاء رابطه‌ای را فراهم می‌کند. در مدل شیء LINQ to SQL، دستورات بانک اطلاعاتی به طور مستقیم به بانک اطلاعات ارسال نمی‌شوند. به عنوان یک توسعه دهنده، به سادگی مقادیر را تغییر می‌دهید و متدها را از طریق این مدل شیء اجرا می‌کنید.

LINQ to SQL، سپس این تغییرات و متدها را به دستورات مناسب SQL ترجمه می‌کند و به بانک اطلاعات می‌فرستد تا اجرا شوند. مدل شیء از طریق ارتباط شیء با بانک اطلاعات کار می‌کند و مدل بانک اطلاعات کارهای دریافت کرده را اجرا می‌کند. در جدول ۱ – ٤ ارتباط بین مدل شیء LINQ to SQL و مدل رابطه‌ای متناظر آن آمده است.

جدول ۱ – ٤ ارتباط بین LINQ to SQL و مدل رابطه‌ای متناظر آن.

شیء LINQ to SQL	شیء رابطه‌ای
DataContext	Database
EntityClass	Table
ClassMember	Column
Assoclation	رابطه Foreign _ Key

۱ – ۴. کلاس DataContext

قبل از این که بتوانید یک پرس‌وجوی LINQ to SQL را اجرا کنید، باید یک اتصال به منبع داده برقرار نمایید. این عمل در LINQ to SQL توسط کلاس DataContext انجام می‌شود. کلاس DataContext، رشته‌ای را به عنوان پارامتر می‌پذیرد. این رشته شامل اطلاعاتی از قبیل نام سرویس دهنده بانک اطلاعات، نام بانک اطلاعات، نام کاربری و کلمه عبور و دیگر اطلاعات است. این اطلاعات در زیر آمده‌اند:

❖ **پارامتر Server (Data Source)**، نام و آدرس سرویس دهنده‌ای را تعیین می‌کند که بانک اطلاعات بـر روی آن نصب شده است. برخـی از مقـادیری کـه ایـن پـارامتر مـی‌پـذیرد، عبـارت‌انـد از: TCI، ١٠,١١١,١,٤ و C:\my.mdb.

❖ **پارامتر Initial Catalog**، نام بانک اطلاعات را نعیین می‌کند.

❖ **پارامتر UID (User ID)**، حساب کاربری را تعیین می‌کند که می‌خواهد به بانک اطلاعاتی وصل گردد.

❖ **پارامتر PWD (PassWord)**، کلمه عبور کاربر را تعیین می‌کند کـه مـی‌خواهـد از طریـق آن بـه بانـک اطلاعات متصل گردد.

❖ **پارامتر Integrated Security**، روش برقراری امنیت را تعیین می‌کند که می‌تواند یکی از مقـادیر true، SSPI (معادل true) و false را بپذیرد.

اکنون دستورات زیر را ببینید:

```
DataContext  db = new  DataContext ("Initial Catalog = Chek ;
Data Source = . \\TCI ; Integrated Security = sspi" ) ;
```

این دستور، نام سرویس دهنده را TCI\\. و نام بانک اطلاعات را Chek در نظر می‌گیرد.

کلاس DataContext دارای متدهای زیر است:

❖ **متد CreateDatabase**، بانک اطلاعات را ایجاد می‌کند.

❖ **متد DeleteDatabase**، بانک اطلاعات را حذف می‌نماید.

❖ **متد DatabaseExists**، تعیین می‌کند آیا بانک اطلاعات وجود دارد یا خیر.

استفاده از صفت Database

صفت Database، نام بانک اطلاعات را در هنگام نگاشت بین یک بانک اطلاعـات و شـیء LINQ تعریـف می‌کند. این صفت خاصیتی به نام Name دارد که نام بانک اطلاعات را تعیین می‌کند. به عنوان مثـال، دسـتورات زیر را ببینید:

```
[ Database  ( Name = "test" )]
  public class testDb
    {
     "
    }
```

این دستورات، نام بانک اطلاعات را test در نظر می‌گیرند و کلاسی به نام testDb ایجاد می‌نمایند.

۱-۱-۴. نوع قوی DataContext

ایجاد یک نوع قوی DataContext خیلی ساده است. برای ایجاد نوع قوی DataContext باید کلاس جدیدی ایجاد کنید که از کلاس DataContext مشتق می‌شود. به عنوان مثال، دستورات زیر را ببینید:

```
public  class chek : DataContext
{
  public chek ( String  connection ) : base (connection){}
  // table   definitions
 …
}
```

این دستورات، یک کلاس DataContext به نام Chek ایجاد می‌کنند که از این کلاس می‌توانید برای اتصال به بانک اطلاعاتی استفاده کنید. نام نوع قوی DataContext هم نام با نام بانک اطلاعاتی است که می‌خواهید به آن متصل شوید. به عنوان مثال، دستورات زیر را ببینید:

```
Chek db = new Chek("Integrated Security=sspi");
bool _dbExists = db.DatabaseExists();
if (_dbExists == true)  textBox\.Text = "Yes, Exists!";
```

دستور اول، شیءای به نام db تعریف می‌کند که به بانک اطلاعاتی Chek وصل می‌شود. دستور دوم، اگر بانک اطلاعاتی (Chek) موجود باشد، به متغیر dbExists_ مقدار true، وگرنه، مقدار false را تخصیص می‌دهد و دستور سوم، اگر مقدار dbExists_ برابر true باشد، در کنترل textBox۱ عبارت "Yes,'Exists!" را قرار می‌دهد. اکنون، دستورات زیر را ببینید:

```
public classChek: DataContext
{
 public Chek(string connection) : base(connection) {}
 public Table<State> State;
}
```

این دستورات، نوع قوی DataContext به نام Chek را تعریف می‌کنند و یک جدول به نام State برای آن تعریف می‌نمایند. این دستورات، معادل دستورات زیر می‌باشند:

```
DataContext context = new DataContext("Initial Catalog=Chek;Integrated
Security=sspi");
Table<State> con = context.GetTable<State>();
```

۲ – ۱ – ۴. نگاشت جداول

جداول بانک اطلاعات توسط کلاس‌های موجودیت در LINQ to SQL نمایش داده می‌شوند. هر کلاس نرمال، به یک جدول بانک اطلاعاتی نگاشت می‌شود (به آن تخصیص می‌یابد).

صفت Table توسط LINQ to SQL، یک کلاس موجودیت را به یک جدول یا دید نگاشت می‌کند. این صفت همچنین یک خاصیت به نام Name دارد که نام جدول یا دید در بانک اطلاعات رابطه‌ای را تعیین می‌کند. به عنوان مثال، دستورات زیر را ببینید:

```
[Table (Name = "db.State") ]
 public class  State
 {
  //
 }
```

این دستورات، جدولی به نام db.State را به یک کلاس به نام State نگاشت می‌کنند.

۳ – ۱ – ۴. نگاشت ستون‌ها

بعد از این که جدول به یک کلاس نگاشت گردید، باید ستون‌های جدول به خواص کلاس نگاشت شود. صفت Column، ستونی از جدول بانک اطلاعات را به اعضای کلاس نگاشت می‌نماید. صفت Column از خواص متعددی تشکیل شده است که در جدول ۲ – ٤ آمده‌اند. به عنوان مثال، دستورات زیر را ببینید:

```
[ Column  (DB Type = "VarChar(٦)", IsPrimarykey =
     true,CanBeNull = false)]
   public  string  StateCode ;
[ Column (DBType = " VarChar (٥٠)", [(CanBeNull = false
public StringState name; )]
```

این دستورات، خواص ستون‌های StateCode و StateName را تعریف می‌کنند که StateCode کلید اصلی است.

۲ – ۴. دستکاری داده‌ها

یکی از بخش‌های بسیار مهم هر بانک اطلاعاتی دستکاری داده‌های آن می‌باشد. یعنی، باید بتوان از طریق LINQ to SQL داده‌ها را دستکاری نمود و نتیجه را به بانک اطلاعاتی برگشت داد. به همین دلیل در این بخش به مفاهیم زیر می‌پردازیم:

۱. اضافه کردن رکورد **۲.** حذف رکورد **۳.** به روز رسانی رکورد

۱ – ۲ – ۴. اضافه کردن رکورد

برای اضافه کردن اطلاعات جدید می‌توانید از متد InsertOnSubmit() کلاس مورد نظر استفاده کنید. این متد، به صورت زیر به کار می‌رود:

InsertOnSubmit (DataContext شیء) ; شیء جدول .

به عنوان مثال، دستورات زیر را ببینید:

```
CityDataContext db = new CityDataContext("Data source=TCI;Initial
Catalog=Chek;Integrated Security=sspi");
Table<City> city = db.GetTable<City>();
City con = new City();
con.cityCode = textBox۱.Text ;
con.cityName = textBox۲.Text;
city.InsertOnSubmit(con);
```

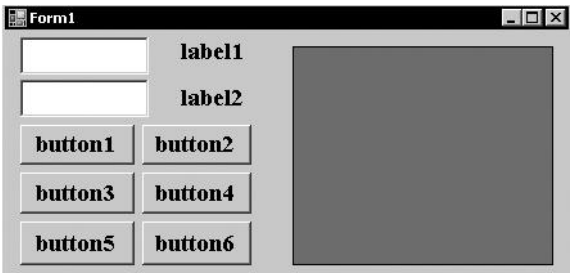
مثال ۱ – ٤ برنامه‌ای که اعمال زیر را بر روی جدول City از طریق LINQ انجام می‌دهد:

- ☒ رکوردی به جدول اضافه می‌کند.
- ☒ رکوردهای جدول را جست‌وجو می‌نماید.
- ☒ رکوردی از جدول را ویرایش می‌کند.
- ☒ رکوردی از جدول را حذف می‌نماید.
- ☒ رکوردهای جدول را نمایش می‌دهد.

مراحل طراحی و اجرا

۱. پروژه جدیدی از نوع Windows Forms ایجاد کنید.

۲. دو کنترل Label (برای نمایش عنوان فیلدهای جدول City)، دو کنترل TextBox (برای دریافت اطلاعات فیلدهای جدول City)، یک کنترل DataGridView (برای نمایش رکوردهای جدول City) و شش کنترل Button (برای انجام عملیات‌های درخواست شده بر روی جدول City) به فرم برنامه اضافه کنید.



۳. کلاس CityDataContext را به پروژه اضافه کنید. برای این منظور، گزینه Project/Add Class را اجرا کنید و در پنجره Add New Item، جلو Name، نـام کـلاس را CityDataContext. Cs وارد کـرده و دکمـه Add را کلیک کنید تا کلاس CityDataContext به پروژه اضافه گردد.

٤. مرجع System.Data.Linq را به پروژه اضافه کنید. برای ایـن منظـور، گزینـه Project/Add Reference را اجرا کرده، از کادری که ظاهر می‌شود، گزینه System.Data.Linq را انتخاب کرده، دکمه OK را کلیک کنید.

۵. به بخش using این کلاس بروید و دستورات زیر را اضافه کنید:

```
using System.Data.Linq;
using System.Data.Linq.Mapping;
```

دستور اول، موجب می‌شود تا بتوانید از کلاس DataContext استفاده کنید و دستور دوم، جهت اسـتفاده از کلاس < نام جدول> Table به کار می‌رود.

٦. دستورات این کلاس را به صورت زیر تغییر دهید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.Linq;
using System.Data.Linq.Mapping;
namespace LINQ
{
    public class CityDataContext : DataContext
    {
        public CityDataContext(string connection) : base(connection) { }
        public Table<City> City;
    }
    [Table(Name = "[dbo].[City]")]
    public class City
       {
        private string _cityCode;
        private string _cityName;
```

```
        [Column(DbType = "varchar(۲۰) ", Storage = "_cityCode", IsPrimaryKey
              = true)]
        public string cityCode
        {
            get
            {
                return _cityCode;
            }
            set
            {
                _cityCode = value;
            }
        }
        [Column(DbType = "varchar(۵۰) ", Storage = "_cityName")]
        public string cityName
        {
            get
            {
                return _cityName;
            }
            set
            {
                _cityName = value;
            }
        }
    }
}
```

این دستورات دو کلاس زیر را تعریف می‌کنند:

☒ **کلاس CityDataContext**: یک نوع قوی تعریف کرده که برای اتصال به بانک اطلاعات به کار می‌رود و جدول City را برای نوع DataContext معرفی می‌نماید.

☒ **کلاس City**: برای تعریف ارتباط جدول بانک اطلاعات و نوع DataContext به کار می‌رود. این کلاس دارای اعضای زیر است:

☒ **فیلد _cityCode**: برای ذخیره ستون (فیلد) cityCode از جدول City به کار می‌رود.

☒ **فیلد _cityName**: برای ذخیره مقدار ستون نام شهر (cityName) از جدول City استفاده می‌شود.

☒ **خاصیت cityCode**، برای بازیابی و مقداردهی مقادیر فیلد _cityCode به کار می‌رود این خاصیت از دو متد get و set استفاده می‌نماید.

☒ **خاصیت cityName**، برای بازیابی و مقداردهی به فیلد cityName_ به کار می‌رود این خاصیت پیاده‌سازی دو متد get وset را دارد.

دستور زیر را مشاهده می‌کنید:

```
[Column(DbType = "varchar(۲۰) ", Storage = "_cityCode", IsPrimaryKey = true)]
```

این دستور، برای تعریف یک ستون جدول به کار می‌رود. (ستون cityCode)، این ستون دارای نوع VarChar(۲۰)، حافظه ذخیره‌سازی به نام cityCode_ و از نوع کلید اولیه (Primary Key) است.

۷. به فرم برنامه برگردید و گزینه View/Code را اجرا کنید تا کد برنامه ظاهر شود. اکنون دستورات زیر را به بخش using اضافه کنید:

```
using System.Data.Linq;
using System.Data.Linq.Mapping;
```

۸. دستورات زیر را به بعد از بلاک بسته (}) مربوط به فراخوانی تابع ()InitializeComponent اضافه کنید:

```
CityDataContext db;
Table<City> city;
```

این دستورات، متغیرهای db و City را از نوع CityDataContext و <City> Table تعریف می‌کنند (متغیر db، برای اتصال به بانک اطلاعات به کار می‌رود و متغیر City برای کار با جدول City استفاده می‌شود).

۹. ناحیه خالی فرم را کلیک مضاعف کنید و دستورات زیر را تایپ نمایید:

```
private void Form۱_Load(object sender, EventArgs e)
 {
   db = new CityDataContext("Data source=RAM\\SQLEXPRESS;Initial Catalog =
        Chek;Integrated Security=sspi");
   city= db.GetTable<City>();
   label۱.Text = "کد شهر";
   label۲.Text = "نام شهر";
   button۱.Text = "اضافه کردن";
   button۲.Text = "جستوجو";
   button۳.Text = "ویرایش";
   button٤.Text = "حذف";
   button٥.Text = "نمایش کل";
   button٦.Text = "خروج";
   button٥_Click(sender, e);
}
```

دستور اول، یک نمونه جدید از نوع CityDataContext ایجاد کرده، آن را به متغیر db تخصیص می‌دهد و خاصیت ConnectionString آن را معرفی می‌نماید. دستور دوم، اطلاعات جدول City از بانک اطلاعات Chek

را با متد GetTable() خوانده، در متغیر city از نوع جدول قرار می‌دهد. دستورات بعدی، خواص کنترل‌های روی فرم را مقداردهی می‌کنند و دستور آخر، رویداد button۵_Click را فراخوانی می‌کند تا اطلاعات جدول City از بانک اطلاعات Chek را بر روی dataGridView۱ نمایش دهد.

۱۰. برنامه را ذخیره و اجرا کنید تا شکل ۱ – ٤ ظاهر شود (در این شکل فرض شده است که جدول City خالی است). در جلوی کد شهر، ۱ و جلوی نام شهر، بابل را وارد نمایید و دکمه **اضافه** را کلیک کنید تا یک پیام ظاهر شود. دکمه Ok را کلیک نمایید تا شکل ۲ – ٤ نمایش داده شود. به همین طریق رکوردهای دیگری را اضافه کنید (شکل ۳ – ٤). اکنون در جلوی کد شهر، مقدار ۳ را وارد کرده، دکمه **جست‌وجو** را کلیک کنید تا شهر با کد ۳ پیدا گردد. نام شهر را به **بابلسر** تغییر دهید و دکمه **ویرایش** را کلیک کنید. اکنون شکل ٤ – ٤ ظاهر می‌شود. در این شکل می‌بینید که نام شهری با کد ۳، به بابلسر تغییر یافته است. برای حذف رکورد فعلی دکمه **حذف** را کلیک کنید تا شهر **بابلسر** حذف شود (شکل ۵ – ٤). برای خروج از برنامه، دکمه **خروج** را کلیک کنید. در این برنامه برای ویرایش و حذف رکورد، ابتدا باید با دکمه **جست‌وجو** آن را پیدا کرده، سپس آن را ویرایش یا حذف نمایید.

شکل ۱ – ۴ نمونه خروجی مثال ۱ – ۴.

شکل ۲ – ۴ نمونه خروجی مثال ۱ – ۴ بعد از اضافه کردن یک رکورد.

شکل ۳– ۴ نمونه خروجی مثال ۱ – ۴ پس از اضافه کردن چند رکورد.

شکل ۴– ۴ نمونه خروجی مثال ۱ – ۴ پس از ویرایش رکورد ساری به بابلسر.

شکل ۵ – ۴ نمونه خروجی مثال ۱ – ۴ پس از حذف رکورد بابلسر.

فصل ۵ قطعه LINQ to Dataset

اکثر برنامه نویسان دات‌نت با مفهوم Dataset آشنا هستند. زیرا، یکی که از مهمترین قطعات ADO.NET است. به زبان ساده، Datasetها اشیایی هستند که شامل جداول داده داخلی توسط کاربر برای استفاده در برنامه کاربردی‌اش در دسترس می‌باشند. Dataset در حافظه نهان محلی ذخیره می‌گردد. این حافظه نهان به کاربران اجازه می‌دهد تا در حالت **بی اتصال** با بانک اطلاعاتی کار کنند. یک Dataset از مجموعه‌ای از جداول (Tables)، ارتباط بین آن‌ها (RelationShips)، سطرها (DataRows)، ستون‌ها (DataColumns)، محدودیت (Constraints) و غیره تشکیل شده است. بنابراین، در این فصل با ارائه یک مثال، به مفاهیم زیر می‌پردازیم:

- دستیابی به داده از طریق ADO.NET
- کلاس‌های ADO.NET
- پرس‌وجو در Datasetها از طریق LINQ to Dataset
- انقیاد داده‌ها
- مقایسه سطرها در Dataset

۱-۵. دستیابی به بانک اطلاعات با ADO.NET

بیان گردیده که بانک اطلاعاتی همان فایل‌های کامپیوتری (فایل‌های داده و کارنامه که در فصل ۲ دیدید) هستند که برنامه کاربردی با استفاده از سیستم مدیریت بانک اطلاعات (DBMS[۳۵]) آن را پردازش می‌کند. امّا، برای این که برنامه کاربردی با سیستم مدیریت بانک اطلاعات ارتباط برقرار کند، نیاز به واسط[۳۶] نرم افزاری دارد. یکی از واسط‌های نرم افزاری بسیار مهم ADO.NET[۳] می‌باشد. این واسط امکان ارتباط با بانک اطلاعاتی

---

۱ .Database Management System ۲.Interface ۳.Activex Data Objects ۴.ConnectionLess

رابطه‌ای و سایر منابع داده را فراهم می‌آورد. برخی از ویژگی‌های ADO.NET از قبیل بی اتصال بودن[٤]، قابلیت برنامه‌نویسی، کارایی، توسعه‌پذیری و غیره، این تکنولوژی را از سایر تکنولوژی‌های دیگر متمایز ساخته است. در این تکنولوژی کلاس‌های متعددی برای کار با بانک‌های اطلاعات مختلف فراهم شده‌اند. برخی از آن‌ها را در ادامه می‌بینید. از آنجایی که در این کتاب بانک اطلاعاتی SQL Server بررسی می‌گردد، به کلاس‌های مربوط به کار با بانک اطلاعاتی SQL Server می‌پردازیم.

۱-۱-۵. کلاس Connection

اولین قدم در عملیات بانک اطلاعاتی، برقراری ارتباط بین برنامه کاربردی و سرویس‌دهنده بانک اطلاعات است. برای این منظور می‌توانید از کلاس Connection استفاده کنید. کلاس‌های مختلفی برای برقراری اتصال با بانک اطلاعاتی به کار می‌روند که برخی از آن‌ها عبارت‌اند از:۱- SqlConnection ۲- oledbConnection ۳- oracleConnection ٤- OdbcConnection و ٥- غیره.

برای استفاده از شیء SqlConnection باید نمونه‌ای از آن ایجاد کرده، پارامترهای رشته اتصال آن را مقداردهی کنید و آن را باز نمایید. پارامترهای رشته اتصال همان پارامترهایی مانند UserID (حساب کاربری)، Password (کلمه عبور)، Data Source (منبع داده)، Initial Catalog (نام بانک اطلاعاتی) و غیره می‌باشند. این پارامترها را در کلاس DataContext دیدید. برای ایجاد نمونه‌ای از شیء SqlConnection به صورت زیر عمل کنید:

```
SqlConnection con = new SqlConnection (" Data Source =
TCI;Intial Catalog = Chek; Integrated Security = SSPI;"
```

این دستور، شیء اتصالی به نام con از نوع SqlConnection تعریف می‌کند. این شیء می‌تواند به بانک اطلاعاتی Chek در سرویس دهنده TCI وصل شود. برای برقراری اتصال (بازکردن اتصال) باید از متد Open() به صورت زیر استفاده کنید:

```
con.Open();
```

این دستور، اتصال به بانک اطلاعاتی را از طریق شیء اتصال con برقرار می‌کند.

بعد از این که کارتان با بانک اطلاعاتی خاتمه یافت، باید اتصال به آن را قطع کنید. برای این منظور می‌توانید از متد Close() به صورت زیر استفاده نمایید:

```
con.Close();
```

۲-۱-۵. کلاس Command

این کلاس برای اجرای دستورات SQL یا رویه‌های ذخیره شده از طریق C# بر روی بانک اطلاعات به کار می‌رود. کلاس‌های متعددی برای Command وجود دارند که برخی از آن‌ها عبارت‌اند از :

۱. کلاس SqlCommand ۲. کلاس OledbCommand ۳. کلاس odbcCommand

۴. کلاس OracleCommand ۵. غیره

برای کار با کلاس SqlCommand، نمونه‌ای از آن ایجاد کرده، خواص آن را مقداردهی نمایید و از متدهای آن استفاده کنید تا دستورات موجود در نمونه بر روی بانک اطلاعات اجرا شوند. برخی از خواص و متدهای مهم کلاس SqlCommand در جدول ۱ – ۵ آمده است.

برای استفاده از شیء SqlCommand مراحل زیر را انجام دهید:

۱. با دستور زیر یک شیء از نوع SqlCommand ایجاد کنید:

```
SqlCommand  cmd = new  SqlCommand();
```

جدول ۱ – ۵ خواص و متدهای مهم شیء SqlCommand.

خاصیت	هدف
CommandText	دستور SQL یا نام رویه ذخیره شده‌ای را تعیین می‌کند که می‌خواهید بر روی بانک اطلاعات اجرا گردد.
CommandType	نوع دستوری را تعیین می‌کند که در CommandText قرار گرفته است.
Connection	شیء اتصالی (Connectionای) را تعیین می‌کند که شیء SqlCommand از طریق آن باید دستورات را اجرا کند.
Parameters	کلکسیون پارامترهای ارسال شده به شیء SqlCommand را تعیین می‌کند.
متد	**هدف**
Cancel	فرمان در حال اجرا را لغو می‌کند.
ExecuteNonQuery	شیء SqlCommand را بر روی بانک اجرا می‌کند و تعداد رکوردهایی را برمی‌گرداند که تحت تاثیر اجرای این دستور قرار گرفته‌اند.
ExecuteScalar	دستور موجود در خاصیت CommandText شیء SqlCommand را اجرا کرده

	اولین فیلد را برمی‌گرداند.
ExecuteReader	دستور موجود در شیء SqlCommand را اجرا کرده، مجموعه‌ای از رکوردها را برمی‌گرداند که قابل ویرایش نیستند (فقط خواندنی هستند).

۲. خاصیت Connection آن را به صورت زیر مقداردهی کنید:

```
Cmd.Connection = con ;
```

con، شیءای از نوع SqlConnection می‌باشد که قبلاً ایجاد گردید.

۳. خاصیت CommandText آن را به صورت زیر مقداردهی نمایید:

```
Cmd.CommandText = "SELECT  *  FROM City" ;
```

این دستور، برای بازیابی اطلاعات جدول City به کار می‌رود. خاصیت CommandText می‌تواند نام رویه ذخیره شده یا نام جدول باشد.

٤. خاصیت CommandType آن را مقداردهی کنید. این خاصیت می‌تواند یکی از مقادیر CommandType.Text (اگر مقدار CommandText یک دستور Sql باشد)، CommandType.StoredProcedure (اگر مقدار خاصیت CommandText نام یک رویه ذخیره شده باشد) و CommandType.TableDirect (اگر مقدار خاصیت CommandText، نام یک جدول باشد) را بپذیرد. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
Cmd.CommandType = CommandType.Text ;
```

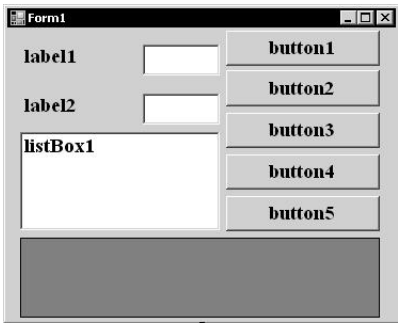
این دستور، نوع دستور cmd را از نوع Text (دستور SQL) در نظر می‌گیرد.

۵. دستور موجود در شیء Cmd را اجرا نمایید. برای این منظور می‌توانید از متدهای ExecuteNonQuery()، ExecuteScalar()، ExecuteReader() و دیگر متدها استفاده کنید. عملکرد این متدها را در جدول ۱ – ۵ دیدید. به عنوان مثال، دستور زیر را ببینید:

```
cmd.ExecuteNonQuery() ;
```

این دستور، دستور موجود در cmd را اجرا کرده اطلاعات جدول City را بازیابی می‌نماید.

مثال ۱ – ۵ برنامه‌ای که روش کار کردن با قطعه LINQ to Dataset را نمایش می‌دهد. این برنامه دارای امکانات زیر است:



- تعداد رکوردهای جداول City و State را می‌شمارد.
- استان‌های که نام آن‌ها با رشته خاصی شروع شوند را نمایش می‌دهد.
- بین جداول State و City ارتباط را ایجاد می‌کند.
- بین جداول City و کنترل DataGirdView انقیاد[۳۷] را برقرار می‌کند.
- دو پرس‌وجو را باهم مقایسه می‌کند.

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام LINQ to Dataset ایجاد کنید.

۲. دوکنترل Label، دو کنترل TextBox، یک کنترل ListBox، یک کنترل DataGridView و پنج کنترل Button به فرم اضافه کنید.

۳. به بخش using برنامه بروید و دستور زیر را اضافه کنید تا بتوانید از کلاس‌های SqlClient استفاده نمایید:

```
using System.Data.SqlClient;
```

٤. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Load آن را به صورت زیر تغییر دهید:

```
private void Form۱_Load(object sender, EventArgs e)
 {
   button۱.Text = "Linq to Dataset";
   button۲.Text = "Single table";
   button۳.Text = "Multiple table";
   button٤.Text = "Data Binding";
   button٥.Text = "Comparing Data";
   label۱.Text = "StartWith ۱";
   label۲.Text = "StartWith ۲";
 }
```

این دستورات، خواص کنترل‌های روی فرم را مقداردهی می‌کنند.

۵. به قبل از رویداد Form۱_Load بروید و دستورات زیر را تایپ کنید:

۱.Binding

```
String strCon="Data Source=TCI;Initial Catalog=Chek;Integrated Security=true";
```

این دستور، رشته اتصال (strCon) را تعریف می‌کند. در این رشته اتصال نام سرویس دهنده بانک اطلاعات (TCI) و نام بانک اطلاعات (Chek) انتخاب گردید و روش احراز هویت را ویندوزی انتخاب می‌کند. یعنی، برای اتصال به بانک اطلاعاتی از کاربر نام کاربری و کلمه عبور را درخواست نمی‌کند.

۶. دکمه button۵ را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

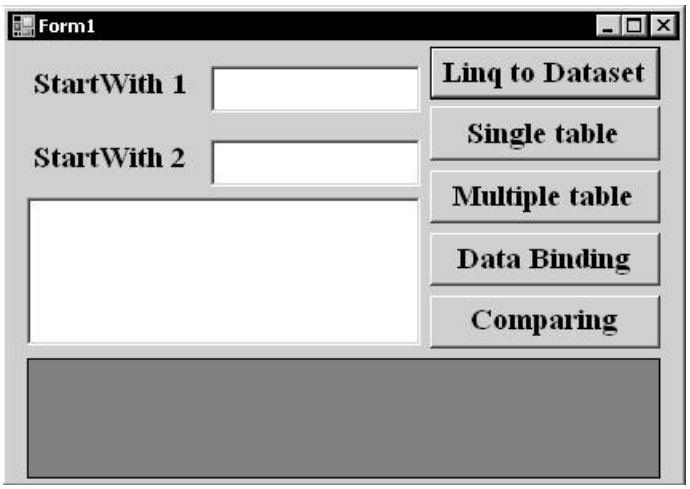
```
private void button۵_Click(object sender, EventArgs e)
 {
   DataSetds = new DataSet();
   string SQL = "SELECT * FROM City";
   SqlDataAdapter da = new SqlDataAdapter(SQL, strCon);
   da.TableMappings.Add("Table", "City");
   da.Fill(ds);
   DataTable header = ds.Tables["City"];
   if (textBox۱.Text != null && textBox۲.Text != null)
   {
      IEnumerable<DataRow> query۱ = from oh in header.AsEnumerable()
        where oh.Field<string>("cityName").StartsWith(textBox۱.Text)
        select oh;
      IEnumerable<DataRow> query۲ = from oh in header.AsEnumerable()
        where oh.Field<string>("cityName").StartsWith(textBox۲.Text)
        select oh;
      if (query۱.Count() != ۰ && query۲.Count() != ۰)
      {
         DataTable dt۱ = query۱.CopyToDataTable<DataRow>();
         DataTable dt۲ = query۲.CopyToDataTable<DataRow>();
         var compareRow = dt۱.AsEnumerable().Intersect(dt۲.AsEnumerable(),
             DataRowComparer.Default);
         listBox۱.Items.Clear();
         foreach (DataRow dr in compareRow)
         {
           listBox۱.Items.Add(dr["cityCode"] + " " + dr["cityName"] + " " +
           dr["StateCode"]);
         }
       }
   }
 }
```

دستورات ۱ تا ٦ مانند رویداد button٤_Click عمل می‌کنند. دستور هفتم، بررسی می‌کند که مقادیر textBox۱ و textBox۲ تهی نباشند. اگر این مقادیر تهی نباشند، دستور هشتم، پرس‌وجوی query۱ را ایجاد

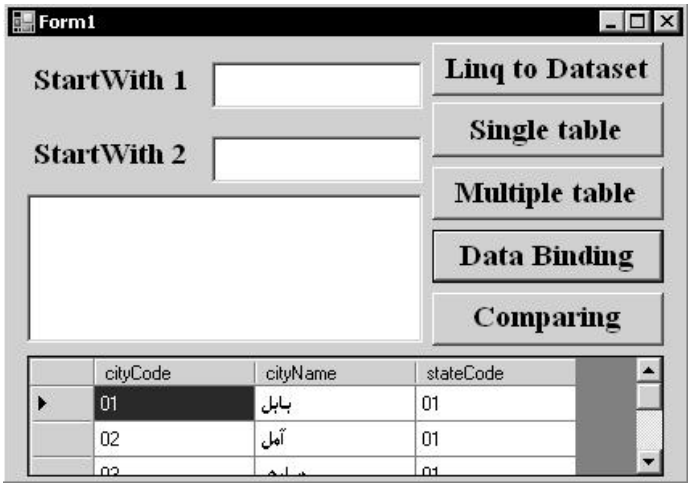
می‌کند. این پرس‌وجو اطلاعاتی از جدول header را بازیابی می‌نماید کـه شـروع نـام شـهر آن در textBox۱ وارد شده باشد. دستور نهم، پرس‌وجوی query۲ را ایجاد می‌کند. این پرس‌وجواطلاعاتی از جدول header را بازیابی می‌نماید که شروع نام شهر آن در کنترل textBox۲ واردشده باشد. دستور دهم، بررسی می‌کند که تعداد رکوردهای پرس‌وجوهای query۱ و query۲ صفر نباشند. اگر تعداد رکوردها صفر نباشد، دستورات یازدهم و دوازدهم، اطلاعات پرس‌وجوهای query۱ و query۲ را در جداول dt۱ و dt۲ کپی می‌کنند. دسـتور سـیزدهم، اطلاعات جداول dt۱ و dt۲ را مقایسه می‌کنند و نتیجه را در پرس‌وجـوی compareRow قـرار مـی‌دهنـد. در پایان، با استفاده از حلقه foreach، اطلاعات پرس‌وجوی compareRow را نمایش می‌دهند.

۷. پروژه را ذخیره و اجرا کنید تا خروجی برنامه ظـاهر شـود (شـکل ۲ – ۵) دکمـه Linq To Dataset را کلیک نمایید تا شکل زیر ظاهر شود:

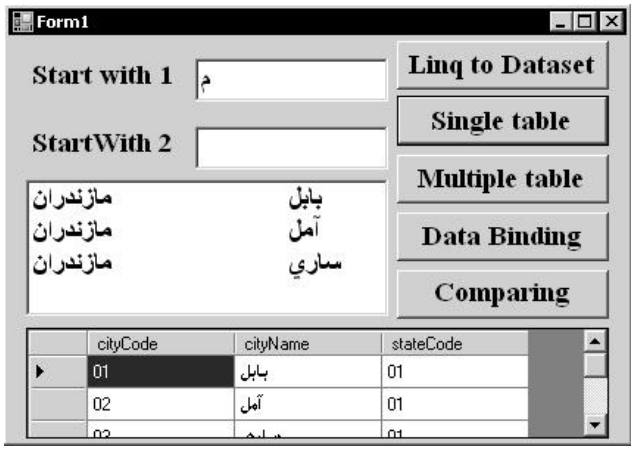
دکمه OK را کلیک کنید. اکنون دکمه Data Binding را کلیک نمایید تا خروجی را به شکل ۳ – ۵ ببینید. دکمه Multiple table را کلیک کرده تا خروجی ٤ – ۵ را مشاهده کنید. در جلوی Start With۱، حرف "م" را وارد کرده، دکمه Single Table را کلیک کنید تا لیست شهرهایی را ببینید که نام استان آن‌ها بـا "م" شـروع می‌شود (شکل ۵ – ۵).



شکل ۲ – ۵ اولین خروجی مثال ۱ – ۵.



شکل ۳ – ۵ نمایش انقیاد جداول به کنترل **DataGridview** .



شکل ۵ ـ ۵ نمایش اطلاعات شهرهایی که نام استان
آن‌ها با " م " شروع می‌شود.