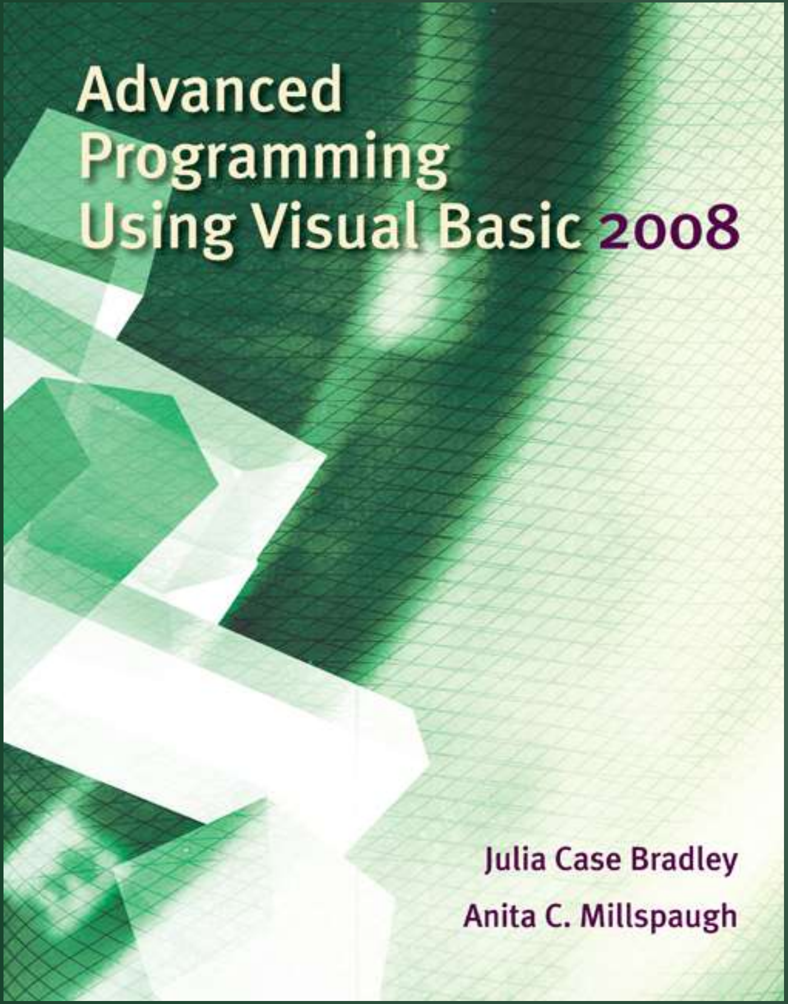


Chapter 13

Additional Topics in Visual Basic

Objectives

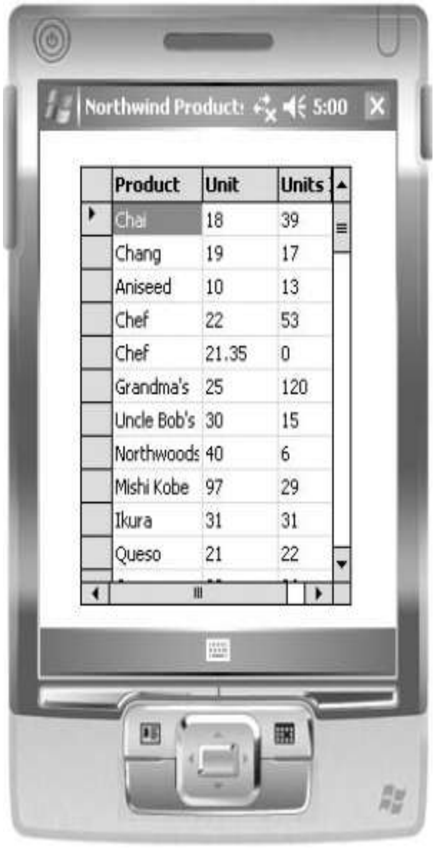
- Write Windows applications that run on mobile devices
- Display database information on a mobile device
- Create interfaces with Windows Presentation Foundation (WPF)
- Query a variety of data sources using Language-Integrated Queries (LINQ)
- Understand and apply the concepts of localization
- Use threading in an application using the BackgroundWorker component

Device Applications

- Creating output for PDAs, cell phones, and pagers requires different protocols
  - Visual Basic can be used to develop applications for these mobile devices
  - The Visual Studio IDE has features for creating solutions that deploy to *Smart Devices*
    - Mobile devices that run compact and mobile versions of Windows

Using Emulators

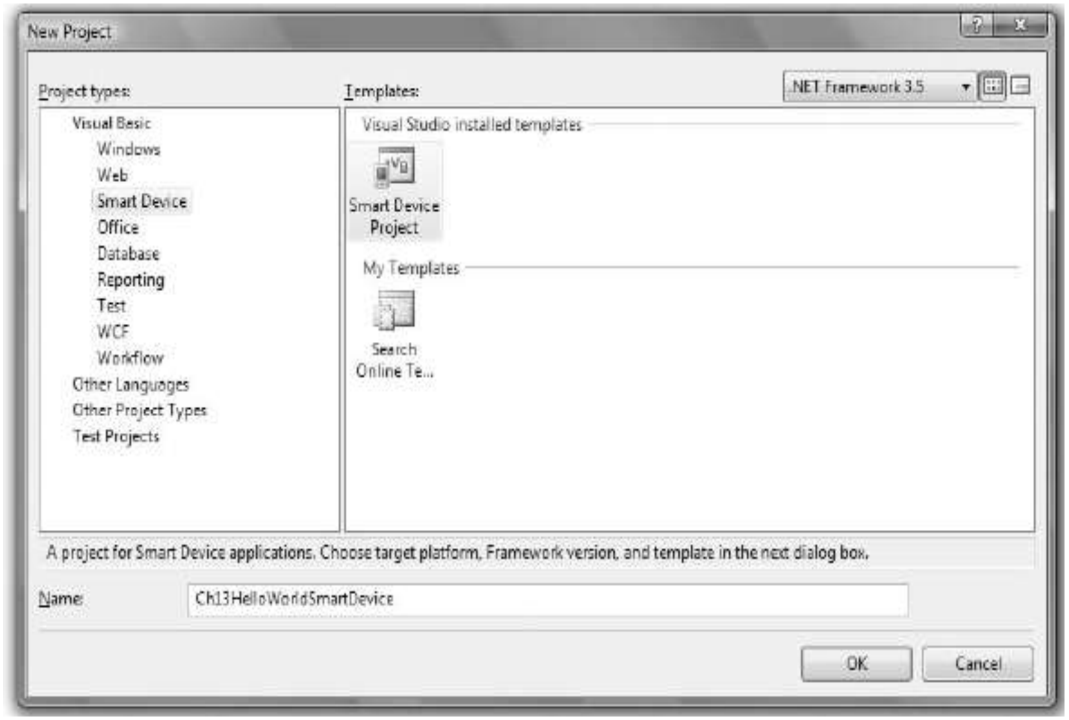
- Smart Device applications developed with VS can be viewed in a regular window or in an *emulator*
  - Emulators provide a better visual concept of the final output
- VS Professional Edition installs several emulators and more are available for download



Product	Unit	Units
Chai	18	39
Chang	19	17
Aniseed	10	13
Chef	22	53
Chef	21.35	0
Grandma's	25	120
Uncle Bob's	30	15
Northwoods	40	6
Mishi Kobe	97	29
Ikura	31	31
Queso	21	22

Smart Device Applications

- Select Smart Device for the project type and the Smart Device Project template

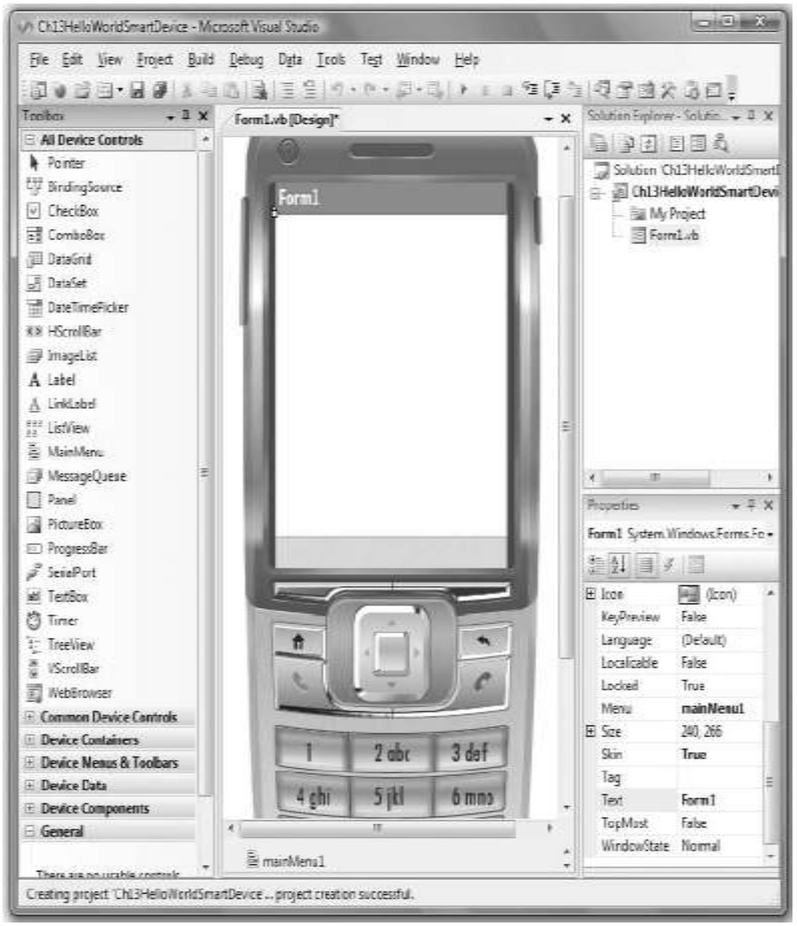


A project for Smart Device applications. Choose target platform, Framework version, and template in the next dialog box.

Name: Ch13HelloWorldSmartDevice

A First Smart Device Application – Step-by-Step - 1

- Create the project
  - Select Smart Device project type
- Add menu items
  - *File*, *Display*, and *Exit*
- Write the code
- Run the application



Form1 System.Windows.Forms.Fo

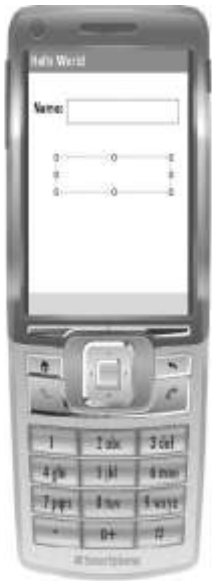
Icon	(Icon)
KeyPreview	False
Language	(Default)
Localizable	False
Locked	True
Menu	mainMenu1
Size	240, 266
Skin	True
Tag	
Text	Form1
TopMost	False
WindowState	Normal

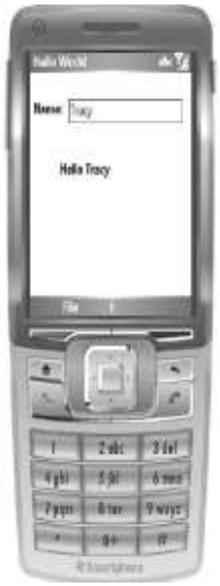
Creating project 'Ch13HelloWorldSmartDevice'... project creation successful.

A First Smart Device Application – Step-by-Step - 2

Add controls to the form

Application running in a device emulator





A Database Application

- Smart Devices can access database files
- Adding database access to a Smart Device application is similar to Windows
- Use the Data Sources window or the *Data* menu
  - Drag a table or fields to the form to create bound controls
- Change the *Data Source* to Microsoft SQL Server Compact 3.5

Changing Column Styles

- Formatting for the columns of a data grid is different from the Windows DataGridView
- The data grid contains a TableStyles property, which is a collection
  - Select the GridColumnStyles collection in the DataGridTableStyle Collection Editor
    - Change the width and header text of individual columns

Creating a Data Form

- The smart tag of the data grid provides the option to *Generate Data Forms*
  - Generates a form with a *New* menu item
    - Delete the *New* menu item and code if not creating a database update program
  - Run a data grid application
    - Double-click on a row in the data grid
    - A single record displays on the form

Updating a Database

- Difficult to test a database update program using an emulator
  - Emulator does not retain the database from one run to the next without advanced configuration
  - If an actual Smart Device is cradled to the computer, transfer the database file to the device and test the update process

Displaying Records in Details View

- Drag the table for a data grid or for a details view
  - No binding navigator
  - Use a combo box
  - In the smart tag, set the combo box DataSource and DisplayMember properties
  - Scroll to the top of the Properties window
    - Expand the *(DataBindings)* entry
    - Select *(Advanced)*
    - Set the *Data Source Update Mode* to Never

Windows Presentation Foundation (WPF) - 1

- Included in Visual Studio 2008
- Create special effects seen in Windows Vista applications
- Write WPF for Windows XP and Vista
  - Special effects do not appear in XP unless a plug-in is installed and the machine is running XP SP2
- Includes development platforms for Windows and Web applications

Windows Presentation Foundation (WPF) - 2

- Write stand-alone and browser applications and programs that display XPS documents
  - Browser applications created through Visual Studio require .NET components to be installed on the client machine and only run in Internet Explorer
- Silverlight is a related technology
  - Has some features of WPF
  - Ability to run on multiple browser platforms
  - More universal development option

The Roles of Designer and Programmer

- Tools make it easy to separate design from programming
  - Programmers generally use Visual Studio
    - Programmer places a button on a window
  - Designers prefer Expression Interactive Designer
    - Designer transforms the button to a flashy design feature
- WPF applications contain two basic files
  - Window.xaml and Application.xaml
  - XAML browser applications are referred to as *XBAP's (XAML* browser applications)

WPF Features - 1

- Feature set includes layouts and controls
  - Controls are very similar to Windows Forms controls
  - Layout is set up in a panel
    - Most common layout is grid
    - DockPanel, Canvas, and StackPanel are other layouts
- XAML style element has same type of functionality as a cascading style sheet
- WPF uses templates, such as data and control templates

WPF Features - 2

- Flexible for including multimedia
  - Include text, documents, images, video, audio, and 2D or 3D graphics
  - Use transformations and effects
    - Rotation and resizing of objects
  - Use a Storyboard class for animation
- WPF includes data binding and interface automation
- Allows creation of hybrid applications
  - Add WPF features to a Windows Forms application or Windows controls to a WPF page

Creating a WPF Project - 1

- Select WPF Application or WPF Browser Application as the project template
  - – Name property is set at the top of the window
  - – A Search box allows a property to be found quickly
  - – Labels have a Content instead of a Text property
  - – A Window has a Title property
  - – A text box has a Text property

Creating a WPF Project - 2

- To change the name of a form
  - Change the name of the file in the Solution Explorer (does not change the name of the class)
  - Use *Refractor/Rename* to change the name of the class
  - If the name of the startup form is changed, change the Startup URI file to the new name

Interoperability

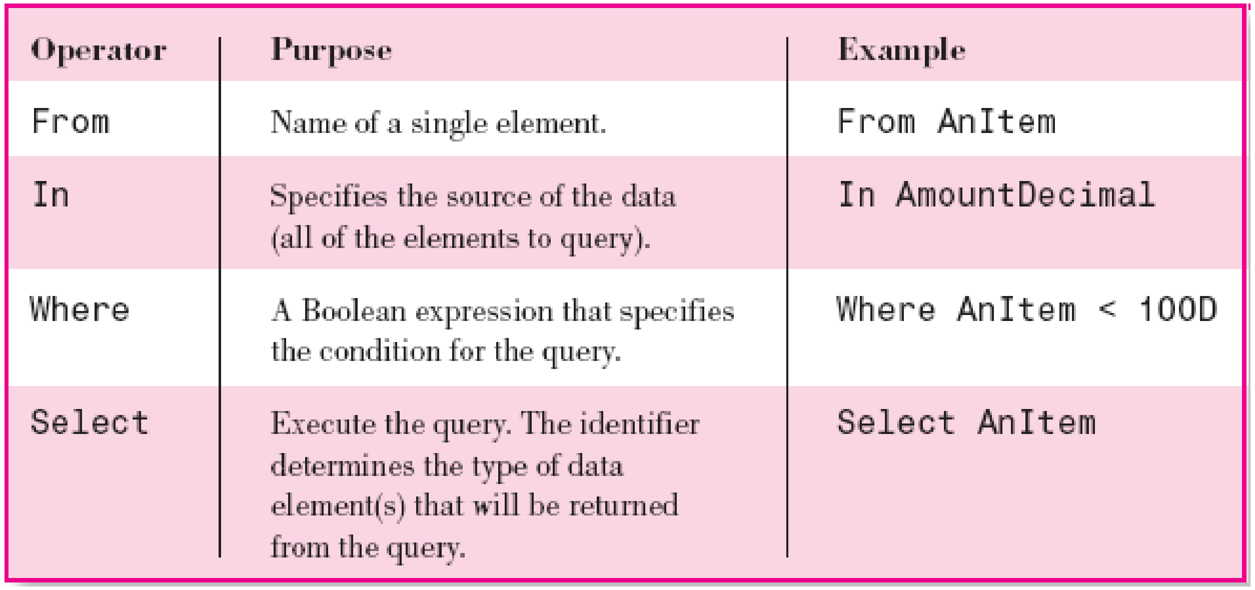
- Toolbox for windows applications contains a *WPF Interoperability* category
  - Contains an ElementHost control
    - A container that allows the addition of WPF controls to a Windows Form
  - Add other WPF controls to the toolbox or add controls in code
    - Add a grid panel inside an ElementHost control
      - Helps lay out multiple controls
    - Add an *Imports* System.Windows.Controls statement to allow the addition of controls in code

LINQ

- A standard language to query any data source defined as an object, a database, or as XML
  - Includes arrays, collections, databases, flat files, and XML

LINQ Keywords - 1

- Operators are standard regardless of source of the data
- Primary LINQ operators
  - *From*, *In*, *Where*, and *Select*



Operator	Purpose	Example
`From`	Name of a single element.	`From AnItem`
`In`	Specifies the source of the data (all of the elements to query).	`In AmountDecimal`
`Where`	A Boolean expression that specifies the condition for the query.	`Where AnItem < 100D`
`Select`	Execute the query. The identifier determines the type of data element(s) that will be returned from the query.	`Select AnItem`

LINQ Keywords - 2

- The LINQ Query – General Form

```
Dim VariableName = From ItemName In Object Where Condition _
    Select ListOfFields/Items
```

  - No data type is specified
  - LINQ uses type inference to allow the *Order By* and *Where* operators to be used on unspecified data types

- The LINQ Query – Example

```
Dim BelowMinimumQuery =
        From AnItem In AmountDecimal
        Where AnItem < 100D
        Select AnItem
```

A First Look at LINQ

- *In* clause refers to the name of the object
- *From* is one element in the collection, does not need to be declared
  - Think of the *From* object as a single element in a *For Each* loop
- *Where* clause allows for a condition
  - If all records are wanted, omit the *Where* clause
- *Select* clause executes the query and gives a result

Additional LINQ Keywords

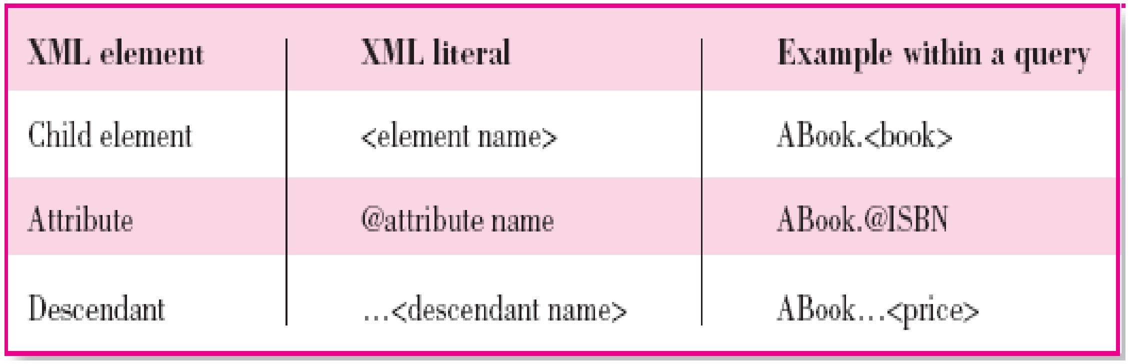
- Operators available for sorting and grouping
- Aggregate operators include *Average, Count, Max, Min,* and *Sum*

LINQ to SQL

- Apply a LINQ query to a relational database
  - Add a LINQ to SQL Classes template to the project
    - Creates a strongly typed DataContext class
  - Use the Object Relational (O/R) Designer
    - Drag database tables from Server Explorer to the design surface
- Refer to the DataContext when writing code

LINQ to XML - 1

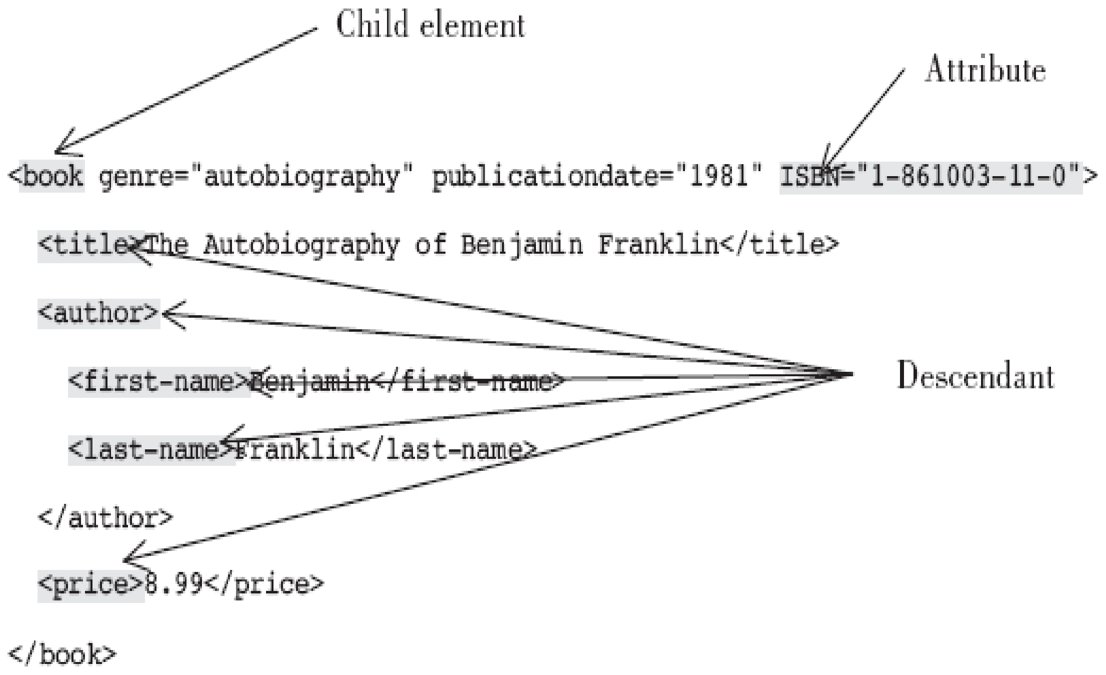
- Retrieve data elements from an XElement or XDocument object
  - Refer to elements in an XElement object in the *In* and/or *Select* clause of LINQ
- *XML literals* simplifies referring to child elements, attributes, and descendants



XML element	XML literal	Example within a query
Child element	<element name>	ABook.<book>
Attribute	@attribute name	ABook.@ISBN
Descendant	...<descendant name>	ABook...<price>

LINQ to XML - 2

- VB three-axis model for referring to elements in a XML document



```
<book genre="autobiography" publicationdate="1981" ISBN="1-861003-11-0">
  <title>The Autobiography of Benjamin Franklin</title>
  <author>
    <first-name>Benjamin</first-name>
    <last-name>Franklin</last-name>
  </author>
  <price>8.99</price>
</book>
```

World-Ready Programs

- *Localization* used to mean creating a separate version of an application for each language or country, after-the-fact
- Today planning of applications used in different countries, languages, and cultures should be part of the original design and development stages

Globalization, Localizability, and Localization - 1

- *Globalization* is process of designing for multiple cultures and locations
  - User interface and output allow for multiple languages
  - Rules and data for a specific language are called a *culture/locale*
    - Contains information about character sets, formatting, currency, measurement rules, and methods of sorting

Globalization, Localizability, and Localization - 2

- *Localizability* determines whether an object can be localized
  - Resources that change are separated from the rest of the code
    - One set of code can change, another set does not change

Globalization, Localizability, and Localization - 3

- *Localization* is the process of translating the interface for a culture/locale
  - Set the form's Localizable property to *true*
    - Set different Text values for each control for each language
  - Form's Language property is set to *Default*
    - The current language set by the operating system
    - Change the Language property to a different language and enter the Text property of each control in that language
      - A separate resource file is created for each language

Globalization, Localizability, and Localization - 4

- The *CultureInfo* class
  - Contains associated language, sublanguage, calendar, and access to cultural conventions
    - Number and date formatting and comparisons of strings
  - Import the System.Globalization namespace to use CultureInfo class

Threading - 1

- A *thread* is a separate execution path that allows a program to do more than one thing at a time
- A program may have several threads running at once
  - Use the BackgroundWorker component to execute time-consuming operations *asynchronously* “in the background”
- The computer switches rapidly from one thread to another, making it appear that all are executing simultaneously

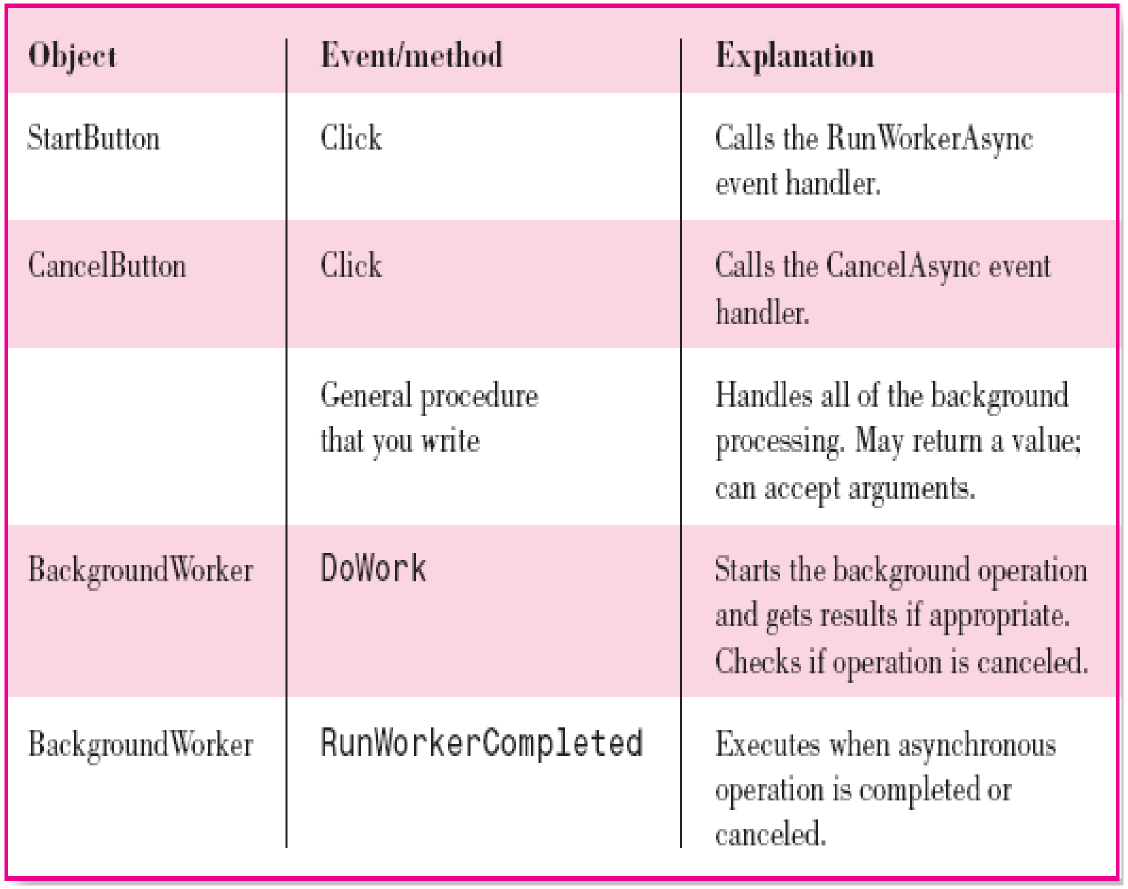
Threading - 2

- *Multitasking* allows the computer to appear as though it is running several programs at once
  - Each program, or *process,* gets a share of the processor time
  - A process requires a complete copy of program code and data
- Within a single program, use *Multithreading* to accomplish multiple tasks
  - Place each task in a separate thread
  - Uses fewer resources because each thread does not require its own copy of code and data
  - Methods that wait for a response are *blocking methods* and are placed in a separate thread so that a problem will interrupt just that thread, not the whole program

Background Workers - 1

- Add a BackgroundWorker component from the *Components* section of the toolbox
- In code, specify which procedure to execute in the background and then call the component's *RunWorkerAsync* method to run the thread
- The *DoWork* method of the Background Worker does the processing
- Background work can be started while executing any procedure
  - Add an *Imports* statement for System.ComponentModel to access the BackgroundWorker class in code

Background Workers - 2



Object	Event/method	Explanation
StartButton	Click	Calls the RunWorkerAsync event handler.
CancelButton	Click	Calls the CancelAsync event handler.
	General procedure that you write	Handles all of the background processing. May return a value; can accept arguments.
BackgroundWorker	DoWork	Starts the background operation and gets results if appropriate. Checks if operation is canceled.
BackgroundWorker	RunWorkerCompleted	Executes when asynchronous operation is completed or canceled.