Set Up the Screen for Your Convenience

- Close or hide extra windows
  - AutoHide feature provides pushpin icon to hide window or “tack” in place
  - Close any window by clicking its Close button
  - Open each window when needed
    - Each window listed on View menu or use buttons on Standard toolbar
- Switch between documents using their tabs or keyboard shortcuts
Use the Full Screen

• Provides maximum screen space
  – Closes all extra windows
  – Hides all toolbars
  – Toggle between full screen and regular display with the Shift + Alt + Enter key combination
Modify the Screen Layout

• Each window in IDE is either a Tool window or a Document window

• Dock windows using the guide diamonds
  – Drag a dockable tool window and diamonds appear to give visual cues to help dock in desired location
Split the Screen Vertically

• View Editor window and form design at the same time
  – Select Window/New Vertical Tab Group
  • Close any extra windows
Set Options for Your Work

- **Projects and Solutions**
  - Set the default folder for projects
- **Text Editor**
  - Set options for all languages or for Basic
  - General, Tabs, and VB Specific
Use Shortcuts in the Form Designer

• Use the Layout toolbar
Nudge Controls into Place

• Hold down the Ctrl key and press one of the arrow keys
  – Moves a control one pixel in specified direction
  – For example, Ctrl + Right arrow moves control one pixel to right
Use Snap Lines to Help Align Controls

- Create or move controls on a form
  - Snap lines pop up to help align controls
    - Blue snap lines appear to align tops, bottoms, lefts or rights of controls
  - Snap lines help standardize vertical spacing between controls
    - Drag control up or down near another control
    - Small dotted line appears to indicate controls are recommended distance apart
Use Shortcuts in the Editor

• Comment and uncomment selected lines

• Use the text editor toolbar
  – Toggle bookmark
  – Jump to next bookmark and Jump to previous bookmark
  – Clear All Bookmarks
Use Keyboard Shortcuts
When Editing Code

• Save time by using keyboard shortcuts
  – Based on the default Visual Basic keyboard mapping
  – Select Tools/Options, Show all settings
  – Select Environment/Keyboard/Keyboard mapping scheme
  – Drop down list and select Visual Basic 6
• Most editing and keyboard shortcuts for Word work in the VS Editor window
Split the Editor Window

- View more than one section of code at a time by splitting the Editor window
Use Drag-and-Drop Editing

- Move or copy text to another location in Editor or another project
  - Move code by selecting text and dragging selection to a new location
  - Copy text by holding down the Ctrl key while dragging
Use the Task List

• Task list displays error messages after a program is compiled
• Add items to the task list as a reminder
  – Write comments in code with the TODO keyword
    ` TODO Come back here and write this code.
    ` TODO Check on this.
• Add tasks by clicking on Create User Task button
  – Creates new line to enter text
Drag Commonly Used Code to the Toolbox

• Select and drag frequently used lines of code
• Insert the code by dragging it from the toolbox to the Editor window
  – Text appears in toolbox only when Editor window is open
Rename Variables and Objects

• Use rename feature to automatically rename variables and objects
  – Right-click a variable or object name
  – Select Rename
  – Enter the new name in the Rename dialog box
  – Item is renamed everywhere in that form
  – For a multiform project, Rename offers the choice of renaming only the current form or all forms
Use Context-Sensitive Help

• Click on a control or a line of code and press F1
  – Help displays closest matching item
• Click on any area of the IDE
  – Press Shift + F1
  – Help will be about the current window or IDE element
Use the Debugging Tools

- Most useful items from the Debug toolbar and menu

<table>
<thead>
<tr>
<th>Menu command or toolbar button</th>
<th>Purpose</th>
<th>Keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Begin debug execution.</td>
<td>F5</td>
</tr>
<tr>
<td>Continue</td>
<td>Continue execution. (Available at break time only.)</td>
<td>F5</td>
</tr>
<tr>
<td>Start Without Debugging</td>
<td>Begin execution without invoking the debugger. This option can make a program run sometimes when it won’t run with Start.</td>
<td>Ctrl + F5</td>
</tr>
<tr>
<td>Stop Debugging</td>
<td>Stop execution of a program.</td>
<td>Shift + F5</td>
</tr>
<tr>
<td>Step Into</td>
<td>Execute the next statement; steps into any called sub procedures or functions. (Available at break time only.)</td>
<td>F8</td>
</tr>
<tr>
<td>Step Over</td>
<td>Execute the next statement; rapidly executes any calls to sub procedures or functions without stepping. (Available at break time only.)</td>
<td>Shift + F8</td>
</tr>
<tr>
<td>Step Out</td>
<td>Rapidly finish the current procedure; reenter break time when the procedure finishes.</td>
<td>Ctrl + Shift + F8</td>
</tr>
</tbody>
</table>
Set Breakpoints

- Breakpoints cause execution to halt on the marked statement
  - Place the mouse pointer in the gray margin indicator area at the left edge of the Editor window and click
    - The line will be highlighted in red
    - A large red displays in the margin indicator area
View the Contents of Expressions

- Point to the variable or expression and pause
  - Current value pops up in a DataTip
- Locals window
  - Displays all objects and variables within scope at break time
- Autos window
  - Displays all variables and control contents referenced in the current statement
  - Also shows the three statements on either side of the current one
Single-Step through Code

• Best way to debug is to thoroughly understand what the project is doing
• Use Visual Studio stepping tools during break time
  – *Step Into*
    • Next line of code executes and program pauses again in break time
  – *Step Over*
    • Displays lines of code in current procedure
    • Does not display lines of code in called procedures
  – *Step Out*
    • Continues rapid execution until called procedure completes
    • Returns to break mode at statement following the *Call*
Write to the Output Window

• Place `Debug.WriteLine` method in code
  – Specify a message to write or an object to track
    `Debug.WriteLine("calculateButton event handler entered.")`
    `Debug.WriteLine(quantityTextBox.Text)`

• Advantage over other debugging techniques is that program execution does not have to break
Copy and Move Projects - 1

• Make sure project is not open
  – Extremely important
• Copy the folder to a new location using Windows Explorer
• Rename the new folder, still using Explorer
• Open the copy in the Visual Studio IDE
• Rename the solution and the project in the IDE’s Solution Explorer
• Rename the forms, if desired
  – If the startup form is renamed, set the Startup Object
• Open the *Project Properties* dialog
  – Change the assembly name to match the new project name
Refresh the Database in an Update Program

- Test all options including adding, deleting, and modifying records
- Delete the modified version of the database
  - Use caution
- Replace the database with an unchanged version
How Windows Applications Handle the Database File

• When setting up a data source, add the database file to the project folder
  – Makes project portable

• When application runs, another copy is made
  – Copy always property makes a new, fresh copy of the file for every program run
  – Copy if newer setting only copies the file if no file exists or the file in bin\Debug is older
  – Return to the original version of the data file by deleting the copy in bin\Debug before running the program
Use Code Snippets and Samples

• Code snippets
  – Small samples of code
  – *Insert Snippet* option on context menu of Editor window
    • Editor places code right in the program

• Sample projects
  – Visual Studio includes many sample projects
    • Use to learn new techniques
  – Walkthroughs in Help give step-by-step introductions to many techniques and controls