.NET Club University Of Cyprus presents:

Microsoft® C#® .NET Crash Course

Creating Windows Applications
Course contents

Overview of the Microsoft .NET Platform

• Lesson 1: Creating a new Windows Application
• Lesson 2: Introduction to Windows Forms
• Lesson 3: Adding Controls to a Form
• Lesson 4: Working with Controls
• Lesson 5: Creating MDI Applications
• Lesson 6: Introduction to Visual Basic
• Lesson 7: Building Mobile Applications
• Lesson 8: Deploying Applications
Course goals

- Create a new Windows application
- Create Windows Forms and add controls to them
- Learn about different types of controls
- Organize controls on a form
- Create MDI (Multiple Document Interface) applications
Overview of the Microsoft .NET Platform
What is the Microsoft .NET Platform

- .NET Framework
- Developer Tools
- Databases
- XML Web Services
- ASP.NET Web Applications
- User Experiences
- Clients
Core Technologies in the .NET Platform

- .NET Framework
- .NET Building Block Services
- Visual Studio .NET
- .NET Enterprise Servers
Components του .NET Framework

Visual Basic®  C++  C#  JScript®  ...

Common Language Specification

ASP.NET: Web Services and Web Forms  Windows Forms

ADO.NET: Data and XML

Base Class Library

Common Language Runtime
Overview of C# .NET

Syntax
Comments

Single-line comments are marked with (//) at the start

//These are single-line comments

Multiple-line comments are marked with

/*
 * These are
 * multi-line comments
 */
Variables

Access Modifiers

public The type or member can be accessed by any other code in the same assembly or another assembly that references it.

private The type or member can only be accessed by code in the same class or struct.

protected The type or member can only be accessed by code in the same class or struct, or in a derived class.

internal The type or member can be accessed by any code in the same assembly, but not from another assembly.

protected internal The type or member can be accessed by any code in the same assembly, or by any derived class in another assembly.
Variables

Variable Declaration

Class Variables

<table>
<thead>
<tr>
<th>Access Modifier</th>
<th>Type</th>
<th>Variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>private</td>
<td>string</td>
<td>name;</td>
</tr>
</tbody>
</table>

Method Variables

- Value Types
  - string name;
  - int x=5;
- Reference Types
  - object obj;
  - object obj=new object();
Variables - Arrays

**Variable Declaration**

**Class Variables**

- Access Modifier: `private`
- Type: `string[]`
- Variable name: `names;`

**Method Variables**

- Value Types
  - `string[] names = new string[4];`
  - `int[] x={1,2,3,4};`
- Reference Types
  - `object[] obj;`
  - `object[] obj=new object[4];`
If statement

```csharp
if (<condition>) {
...
}
else if {
...
}
else {
...
}
```

**Single line if statement**

```csharp
if (<condition>)
```
For statement

FOR statement
for (int I = 1; I<=10; I++) {
    for (int J = 1; I<=10; J=J+2) {
        for (int K = 10 K>=1; K--) {
            // Statements to operate with current values of I, J, and K.
        }
    }
}

FOR each statement
bool Found = false;
List<Object> objectList = new List<Object>();
foreach (Object obj in objectList) { // Iterate through elements.
    if (obj.Name == "Hello") { // If Name equals "Hello"
        Found = true; // Set Found to True.
        break; // Exit loop.
    }
    else continue; }

While and Do Loop statements

While statement
int Counter = 0;
while (Counter < 20) {  // Test value of Counter.
    //Your code
    Counter++;  // Increment Counter.
    // break;  // Exit loop.
}  // End While loop when Counter > 19.

DO DoLOOP statement
    [ statements ]
    [ break ]
    [ statements ]
While condition
Select Case (switch)

Using using System.Diagnostics;
int Number = 4;
// ...
switch(Number) {   // Evaluate Number.
    case 1:   // Number between 1 and 3, inclusive.
        Debug.WriteLine("Between 1 and 3");
        break;
    case 2:
    case 3:
        Debug.WriteLine("Between 1 and 3");
        break;
    case 4:
        Debug.WriteLine("Number 4");
        break;
    default:   // Other values.
        Debug.WriteLine("Not between 1 and 4");
}
Void Methods

Void Methods do not return a value

```csharp
void ComputeArea(double Length, double Width) {
    double Area;   // Declare local variable.
    if (Length == 0 || Width == 0) {
    } else {
        Area = Length * Width;   // Calculate area of rectangle.
        Debug.WriteLine(Area);   // Print Area to Immediate window.
    }
}
```
Non-Void Methods return a value

Function Name: CalcSum
Parameters: (double[] Args)

```csharp
    double CalcSum(double[] Args)
    {
        int I;
        double sum = 0;
        if (Args.Length > 0) // Check if arguments passed.
            for (I = 0; I < Args.Length; I++)
                {
                    sum += Args[I];
                }
        return sum; // Returns latest value of sum.
    }
```
Mechanisms for Passing Parameters

Three ways to pass parameters

- **in** Pass by value
- **in** Pass by reference
- **out** Output parameters
Pass by Value

Default mechanism for passing parameters:
- Parameter value is copied
- Variable can be changed inside the method
- Has no effect on value outside the method
- Parameter must be of the same type or compatible type

```csharp
static void AddOne(int x)
{
    x++; // Increment x
}
static void Main()
{
    int k = 6;
    AddOne(k);
    Console.WriteLine(k); // Display the value 6, not 7
}
```
Pass by Reference

Reference parameters are references to memory locations

Using reference parameters

Use the `ref` keyword in method declaration and call

Match types and variable values

Changes made in the method affect the caller

Assign parameter value before calling the method

```csharp
static void AddOne(ref int x)
{
    x++; // Increment x
}
static void Main()
{
    int k = 6;
    AddOne(ref k);
    Console.WriteLine(k); // Display the value 7
}
```
Output Parameters

Output parameters: Values are passed out but not in
Using output parameters

Like ref, but values are not passed into the method

Use out keyword in method declaration and call

```csharp
static void AddOne(out int x)
{
    x++; // This produces an error
    x=5;
}
static void Main()
{
    int k = 6;
    AddOne(out k);
    Console.WriteLine(k); // Display the value 5
}
```
Lesson 1

Creating a new C# Windows Application
Creating a new C# Project

[Image of the New Project dialog box in Visual Studio, showing project templates for Windows Forms Application, Class Library, WPF Application, etc., with a dialog for specifying project name, location, and solution name.]
Visual Studio .NET Windows

View the solution/project files

Manage Database/Server connections

View controls that can be added in forms and project
Lesson 2

Introduction to Windows Forms
Forms in Solution Explorer

Maintaining two separate files for the design and code simplifies form development.

Form/Class name

Partial Class with all design properties
Form Life Cycle

1. Load
2. Activated
3. FormClosing
4. FormClosed
5. Deactivate
How to handle Form Events

Double click on a form to view the Load and other events
How to set a Form’s Properties

Categorized View

Alphabetical View
TODO: Set the following properties on Form1

To build and start your project:

- Press F5 or
- Press the Play ( ) button

Set the following properties and build your project each time:

- Set MaximizeBox to False
- Set FormBorderStyle to Fixed3D
- Set Size Width to 150
- Set Size Height to 150
- Set WindowState to Maximized
- Set Text to MyFirstForm
Lesson 3

Adding Controls to a Form
How to add Controls to a Form

Open the Toolbox
With a Form open:

- Double click on a control

- Click on the control and
  - Move the mouse over the form
  - Click and drag to create the desired size of the control
Open Form1’s design and
• Add 1 TextBox
  – Set the (Name) Property to txtFirstNumber
  – Set the Text property to “Add first number”
• Add 1 ComboBox
  – Set the (Name) Property to cboOperation
  – Set the Text property to “”
Practice (continued)

- Add 1 TextBox
  - Set the (Name) Property to txtSecondNumber
  - Set the Text property to “Add second number”
- Add 1 Label
  - Set the (Name) Property to lblEquals
  - Set the Text property to “=”
Practice (continued)

• Add 1 TextBox
  – Set the (Name) Property to txtResult
  – Set the Text property to ""

• Add 1 Button
  – Set the (Name) Property to btnCalculate
  – Set the Text property to "Calculate"
Next we need to resize each control
(hint: resize one control and use the Format menu to make each other control follow the same properties)

- Set the Form1’s Size Property to 160,192
- Set each control’s Size Property to 136,20
  - To resize Labels you need to set the AutoSize property to False
- Set each control’s Location→X Property to 8
Practice (continued)

- Set the Location → Y Property to:
  - txtFirstNumber: 8
  - cboOperation: 31
  - txtSecondNumber: 55
  - lblEquals: 78
  - txtResult: 104
  - btnCalculate: 127
Practice (continued)

When you have finished your Form1 should look like this
Let’s talk about controls

- TextBox
- Label
- MainMenu
- CheckBox
- Radio Button
- Group Box
- List Box
- Checked List Box
- Combo Box
- Tree View
- Tab Control
- ...and so much more!!!
Using the Format Menu

Use the format menu to automatically align/resize controls.

After selecting multiple controls, single click on the leader control that will be used to format all other controls.

The leader control is marked with white squares.
Lesson 4

Working with Controls
What is an event?

- Event is a message sent by an object to signal the occurrence of an action.
- An example event is when clicking a button which triggers the (Click event) called Click.
- Other events (e.g., for a TextBox):
  - Click
  - TextChanged
  - LostFocus
  - MouseHover
  - ...

06/10/2011
private void btnCalculate_Click(
    object sender,
    EventArgs e) {

    ... MessageBox.Show("This button will perform the calculation");

}
Practices

• A simple example
• Build a simple calculator
• Working with various controls
Practice: A simple example

Create a new form
To do this:
- Open Solution Explorer
- right click on your project (CSharpCrashCourse)
- Select Add ➔ Add Windows Form
- Name the Form frmSimpleExample
- Add a TextBox and a Button to frmSimpleExample
Practice (continued)

We need to change the name of the Button each time we click on it and set it to whatever the user writes in the TextBox.

Firstly we need to configure the project to start from frmSimpleExample.

To do this right click on the project and select properties.

In the Startup object select frmSimpleExample.
Next we need write code to the Click event of the Button1

- Double Click on Button1

- We need to change the Button1’s Text Property to whatever is written in the TextBox1’s Text.

- Button1.Text = TextBox1.Text;

- Run your project
Questions
Practice: Build a simple calculator

Open Form1
cboOperations must contain the basic mathematical operations
- +
- -
- /
- *

Use the Items property of cboOperations to add the symbols above
Practice: Continued

![Image of a form with a text box labeled "Add the first number" and a calculate button]
Practice: Continued

We assume that the user input will be integers. (next we are going to check if they really are!)
So when we press the button we need to produce a result, according to the cboOperations selected operation, and display it to the user via the txtResult
Practice: Continued

Inside the btnCalculate_Click

We need to assign the values of the two TextBoxes to 2 integer variables.

• Let’s name the a, b
• int a = int.Parse(txtFirstNumber.Text);
• int b = int.Parse(txtSecondNumber.Text);
Next we need to check what operation is selected in cboOperation and calculate the result accordingly.

To get the comboBox selected item use the SelectedItem Property.

- `switch(cboOperation.SelectedItem.ToString())`
- `case "+":`
- `txtResult.Text = a + b;`
- `...`
Now, we want that each time the user clicks on one of the input textboxes to set the Text Property to "" (empty)

We need to add code to the Click event

```csharp
private void txtFirstNumber_Click(object sender, EventArgs e) {
    txtFirstNumber.Text = string.empty;
}

private void txtSecondNumber_Click(object sender, EventArgs e) {
    txtSecondNumber.Text = string.empty;
}
```
Practice: Continued

Can we use one function to handle both TextBox Click events?

Yes

private void txtNumber_Click(object sender, EventArgs e) {
    //The sender is either txtFirstNumber or txtSecondNumber
    System.Windows.Forms.TextBox txt = (TextBox)sender;
    txt.Text = string.Empty;
}
What if the user enters String values in the two textboxes and not Integers???

**Try...Catch**

```csharp
int a, b;
try {
    a = int.Parse(txtFirstNumber.Text);
}
catch (Exception ex) {
    MessageBox.Show(...);
    txtFirstNumber.Focus();
    txtFirstNumber.SelectAll();
}
```
Questions
Practice: Working with various controls
Practice: Continued

Create a new Form (frmWorkingWithControls)
• Change its **Name** Property to **frmWorkingWithControls**
• Change its **Text** Property to **Working With Controls**
• Change its **Size** Property to **518, 494**
• Change its **MinimizeBox** and **MaximizeBox** to **False**
• Change its **FormBorderStyle** Property to **FixedSingle**

To set frmWorkingWithControls as the startup object
• Open Solution Explorer
• Right-Click on project CSharpCrashCourse
• Select Properties
• Under Startup Object select frmWorkingWithControls
Practice: Continued

Create a 5 CheckBoxes

- Change their **(Name)** Property to `chkRed, chkGreen, chkBlue, chkYellow, chkBlack`
- Change their Text Property to **Red, Green, Blue, Yellow, Black** accordingly
- Change their **Size** Property to **104, 24**
- Change their **Location** Property to
  - `chkRed` 8, 32
  - `chkGreen` 8, 64
  - `chkBlue` 8, 96
  - `chkYellow` 8, 128
  - `chkBlack` 8, 160
- You may change their BackColor Property to match each CheckBox’s Name
Create a 2 RadioButtons

- Change their **Name** Property to `rdbMale, rdbFemale`
- Change their Text Property to **Male, Female** accordingly
- Change their **Size** Property to 64, 24
- Change their **Location** Property to
  - `rdbMale` 136, 32
  - `rdbFemale` 136, 64
Practice: Continued

Create a 1 ListBox

- Change its **Name** Property to **lstCity**
- Click on the Items Property, (collection) button and add the following values
  - Leykosa
  - Larnaka
  - Lemesos
  - Pafos
  - Ammohostos
- Change its **Size** Property to **120, 82**
- Change its **Location** Property to **224, 32**
Create a 1 DateTimePicker
- Change its (Name) Property to dtpDateOfBirth
- Change its Size Property to 152, 20
- Change its Location Property to 352, 32

Create 1 Label
- Change its (Name) Property to lblChoose
- Change its Size Property to 496, 24
- Change its Location Property to 8, 8
- Change its Text Property to
  Choose your colors  Gender  City
  Date Of Birth
Practice: Continued

Create a 1 Button
- Change its (Name) Property to btnInfo
- Change its Text Property to Summary
- Change its Size Property to 496, 23
- Change its Location Property to 8, 192

Create 1 TextBox
- Change its (Name) Property to txtInfo
- Change its Multiline Property to True
- Change its Size Property to 496, 232
- Change its Location Property to 8, 224
- Change its Text Property to ""
Practice: Continued

Now we need to add code to btnInfo.Click event to display what we have selected in the other controls to the txtInfo

Double click on btnInfo

We need one variable to store the result.

string summary = “Your favorite colors are:”;

string colors;

int i = 0;
Practice: Continued

We need to check each CheckBox and if it is Checked then we need to add it to the favorite colors.

- To check if a CheckBox is Checked use the **Checked** Property
  - if (CheckBox1.Checked==true) ..., or
  - if (CheckBox1.Checked)

```csharp
if (chkRed.Checked) {
    if (i == 0)
        colors = chkRed.Text;
    else
        colors = colors + "," + chkRed.Text;
    i += 1;
}
```
Wouldn’t it be nice if we could say For each checkbox do ...? Think again → with C# .NET WE CAN!!!

List<Control> controlList = new List<Control>();
    controlList.Add(chkRed);
    controlList.Add(chkGreen);
    controlList.Add(chkBlue);
    controlList.Add(chkYellow);
    controlList.Add(chkBlack);
foreach (System.Windows.Forms.CheckBox checkbox in controlList) {
    if (checkbox.Checked) {
        if (i == 0)
            colors = checkbox.Text;
        else
            colors = colors + "," + checkbox.Text;
        i++;
    }
}
if (i == 0) colors = "None!";

summary += colors;
summary += "\nAnd your gender is:"
    if (rdbMale.Checked)
        summary += "Male";
    else
        summary += "Female";

summary += "\nAnd your city is:" + lstCity.SelectedItem.ToString();

summary += "\nAnd your birthdate is:" + dtpDateOfBirth.Value.ToShortDateString();

txtInfo.Text = summary;
Practice: Continued

Run your project and test it
Questions
Lesson 5

Creating MDI Applications
SDI vs. MDI Applications

**SDI**
- Only one document is visible
- You must close one document before you open another

**MDI**
- Displays multiple documents at the same time
- Each document is displayed in its own window
How to Create MDI Applications

First of all we need to specify a parent form
- Add two new Forms to your project and name them frmParent and frmChild.
- Set frmParent as your project’s startup object
- Open frmParent’s design and set its IsMdiContainer to True

Let’s add a menu to frmParent
- Open the Toolbox
- Double-Click on MenuStrip
- Add the first menu item and name it &File (&→Alt Shortcut)
- Under File add another menu item and name it &New
How to Create MDI Applications (continued)

- Add the second menu item next to File and name it **&Window**
- Under **Window** add the following menu items
  - **Tile &Vertical**
  - **Tile &Horizontal**
  - **&Cascade**
- Set the **MdiWindowListItem** property of the **MenuStrip** to **WindowToolStripMenuItem**
Create a new Child Form

To call a child form from the parent form

- Double Click on menu item New
- Add the following code

```csharp
frmChild newChild = new frmChild();
// Set the parent form
newChild.MdiParent = this;
// Display the form
newChild.Show();
```
Organizing child forms on the parent form

To organize child forms on the parent form

- Double Click on menu item Tile Vertical
  
  this.LayoutMdi(MdiLayout.TileVertical);

- Double Click on menu item Tile Horizontal
  
  this.LayoutMdi(MdiLayout.TileHorizontal);

- Double Click on menu item Cascade
  
  this.LayoutMdi(MdiLayout.Cascade);
Build and run your project

Build and Run your project
Create 3 new child forms and try the Tile and Cascade menu items under windows
Lesson 6

VB.NET → C#
# C# to VB.NET comparison

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. int i;</td>
<td>1. Dim i As Integer</td>
</tr>
<tr>
<td>2. int array[6];</td>
<td>2. Dim array(5) As Integer</td>
</tr>
<tr>
<td>3. for(i=0;i&lt;6;i++){...}</td>
<td>3. For i=0 To 5...Next I</td>
</tr>
<tr>
<td>4. while(cond){...}</td>
<td>4. While &lt;cond&gt;...End While</td>
</tr>
<tr>
<td>5. private void test()...</td>
<td>5. Private Sub test()...</td>
</tr>
<tr>
<td>6. private int test() ...</td>
<td>6. Private Function test() As Integer ...</td>
</tr>
<tr>
<td>7. return/break;</td>
<td>7. Exit ...</td>
</tr>
</tbody>
</table>
Practice

Interacting with Visual Basic and C#
Practice

Create a new Visual Basic Windows Application
Add a new C# Class Library
• Right click on your Solution and Select Add→New Project
• Select C# Project Type
• Select Class Library from the Template
Practice

In the ClassLibrary1 project, edit Class1.cs and create a new method

- Method name: add
- Input Parameters: int a, int b
- Return type: int
- Your code should look like this

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ClassLibrary1
{
    public class Class1
    {
        public int add(int a, int b)
        {
            return a + b;
        }
    }
}
```

Build ClassLibrary1
Practice

Add a reference to ClassLibrary1 from your Windows Application

- Right Click on WindowsApplication1 project and choose Add Reference
- From the Projects tab, select ClassLibrary1
- Review your WindowsApplication1’s references
Practice

In your WindowsApplication1 do the following:

- Add a button in Form1 (button1)
- Double-click on the button and create a new object of ClassLibrary1.Class1
  
  Dim obj As New ClassLibrary1.Class1

- Initialize two integers a and b with values 3, 5
  
  Dim a As Integer = 3
  Dim b As Integer = 5

- Create a messagebox that displays the added value of a and b by calling the add method of obj
  
  MessageBox.Show(obj.add(a, b).ToString)
Questions
Lesson 7
Building Mobile Applications
Create a new Smart Device Project
Select Device Application for building a windows mobile application

Select the correct SDK suitable for your device
Practice

Run all previous practices on the Smart Device Project
Lesson 8

Deploying Applications
Overview

1. Describing Assemblies
   • Assemblies Overview
   • Benefits of Strong-Named Assemblies
   • Creating Strong-Named Assemblies
   • Versioning Strong-Named Assemblies
   • Using the Global Assembly Cache

2. Choosing a Deployment Strategy
   • Deployment Overview
   • Copying Projects
   • Deploying Projects
   • Types of Deployment Projects

3. Deploying Applications
Assemblies Overview

- Contains code, resources, and metadata
- Provides security, type, and reference scope
- Forms a deployment unit
- Versionable
- Side-by-side execution allows multiple installed versions
- Global assembly cache allows assembly sharing
Benefits of Strong-Named Assemblies

• Guaranteed uniqueness
  • No two strong names can be the same
• Protected version lineage
  • Only legitimate assembly versions can be loaded
• Enforced assembly integrity
  • Assemblies are tested for unauthorized modification before loading
Creating Strong-Named Assemblies

Requires identity, public key, and digital signature
Generating the public-private key pair
Create a .snk file
Modify AssemblyInfo.vb

<Assembly: AssemblyKeyFile("KeyFile.snk")>
Versioning Strong-Named Assemblies

When a client makes a binding request, the runtime checks:

- The original binding information inside the assembly
- Configuration files for version policy instructions

Use a publisher policy file to redirect the binding request

```xml
<bindingRedirect oldVersion="1.0.0.0" newVersion="1.0.1.0"/>
```
Using the Global Assembly Cache

- **Performance**
  - Quicker binding
  - Only one instance ever loaded
- **Shared location**
  - Can use machine configuration file to redirect bindings
- **File security**
  - Only administrators can delete files
- **Side-by-side versioning**
  - Can install multiple copies using different version information
Deployment Overview

- No-impact applications
- Private components
- Side-by-side versioning
- XCOPY deployment
- On-the-fly updates
- Global assembly cache
Copying Projects

- Copying a project
  - There is an extra menu command for Web applications
  - You can copy a project directly to a Web server
- Using the XCOPY command
  - Use the DOS command
  - You can use it for any type of application
Deploying Projects

- Windows Installer
  - Is used for Windows-based and Web-based deployment
  - Copies all required files and registers components
  - Configures IIS for Web-based applications
- Merge modules
  - Are used for reusable components
  - Are included in an .msi file
Types of Deployment Projects

- Cab Project
  for downloading from Web server
- Merge Module Project
  for shared components
- Setup Project
  for Windows-based applications
- Setup Wizard
  to walk through deployment project creation
- Web Setup Project
  for Web-based applications
Deploying Applications

- Creating a Merge Module Project
- Creating a Setup Project
- Using the Editors
- Creating Installation Components
- Deploying the Application
Creating a Merge Module Project

- Never installed directly included in client application deployment project
- Contains
  - DLL
  - Any dependencies and resources
  - Information for the Windows Installer
- Store one component only in one merge module
  If component changes, create a new merge module
Creating a Setup Project

Creates a blank setup project with relevant folders in File System Editor

- **Windows-based application**
  - Application Folder
  - User’s Desktop
  - User’s Programs Menu

- **Web-based application**
  - Web Application Folder
<table>
<thead>
<tr>
<th>Using the Editors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registry</td>
</tr>
<tr>
<td>File types</td>
</tr>
<tr>
<td>User interface</td>
</tr>
<tr>
<td>Custom actions</td>
</tr>
<tr>
<td>Launch conditions</td>
</tr>
</tbody>
</table>

Creating Installation Components

EventLog
MessageQueue
PerformanceCounter
Service
ServiceProcess
<table>
<thead>
<tr>
<th>Deploying the Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows-based setup project</td>
</tr>
<tr>
<td>Copies all files</td>
</tr>
<tr>
<td>Registers components</td>
</tr>
<tr>
<td>Performs other installation tasks</td>
</tr>
<tr>
<td>Web setup project</td>
</tr>
<tr>
<td>Copies all files</td>
</tr>
<tr>
<td>Registers components</td>
</tr>
<tr>
<td>Creates a Web application</td>
</tr>
</tbody>
</table>
Practice

Create a setup project for a windows application
Add a new Setup Project to your solution
Select the Setup Wizard template
Choose a project type

The type of project determines where and how files will be installed on a target computer.

Do you want to create a setup program to install an application?

- Create a setup for a Windows application
- Create a setup for a web application

Do you want to create a redistributable package?

- Create a merge module for Windows Installer
- Create a downloadable CAB file
Practice

Press Finish when you are done

This is the .exe file of your application
Practice

Build your project

Using explorer, go to 
%Solution Folder%\Setup1\Debug

Check out the files
- Setup1.msi
- setup.exe

Run setup.exe and install your application
Practice

Creating a shortcut to Desktop of your application

Create a Shortcut to Primary Output

Move it on the User’s Desktop Folder
Lesson 9

Miscellaneous Topics

System.IO
System.Drawing
System.Diagnostics.Process
System.IO: File/Directory Operations

**IO.File: File Operations**
- Copy, Delete, Exists, Move, Replace, ...
- Open(path, FileMode), Close,...

**IO.FileMode: File mode when opening a file**
- Append, Create, CreateNew, Open, OpenOrCreate, Truncate

**IO.Stream**
- FileStream, StreamReader/Writer, TextReader/Writer,...

**IO.Directory: Directory Operations**
- CreateDirectory, Exists, Move, ...
- GetDirectories(path), GetFiles(path), GetParent(path)
System.IO: File/Directory Operations

**Example:** Reading all files in a directory

```csharp
// Directory Path
string dirPath = "c:\tmp";

// stream reader for each file
StreamReader sr = null;

// String for reading each line of the file
string strLine = string.Empty;

foreach (string file in Directory.GetFiles(dirPath)) {
    // Create a new StreamReader for this file
    sr = new StreamReader(file);

    // OPTION 1: read line by line
    while (!sr.EndOfStream)
        strLine = sr.ReadLine();

    // OPTION 2: read all
    strLine = sr.ReadToEnd();

    // Close stream reader (otherwise file is locked)
    sr.Close();
}
```
System.Drawing: Drawing custom shapes

System.Drawing.Graphics: graphic objects used for drawing
CreateGraphics method: gets the graphics object for a control
Draw methods: Draw the outline of a shape (e.g., DrawRectangle)
Fill methods: Draw the fill of a shape (e.g., Fill Rectangle)

Example: Drawing a custom rectangle on a Form

```
// Create a graphics object
Graphics g;

// Get the Forms graphics object
g = this.CreateGraphics();

// Draw the outline of a rectangle
g.DrawRectangle(Pens.Black, _
50, 20, 100, 200);
X, Y, Width, Height
```
System.Diagnostics.Process

To launch an external process (e.g., Windows Calculator)

Thanks for your participation