MS C# Programming

Chapter 1: Introduction

Lesson 1

.NET Platform and Visual Studio
Objectives

You will learn:

• The relationship between C# and .NET.
• The components of a windows program.
• The components of the .NET Framework.
• The components of a C# program.

C# Features

• C# facilitates the development of software components through several innovative language constructs.
  – Encapsulated method signatures, known as delegates, which enable type-safe event notifications.
  – Properties, which serve as accessors for private member variables.
  – Attributes, which provide declarative metadata about types at run time.
  – Inline XML documentation comments.
  – LINQ - Language-Integrated Query which provides built-in query capabilities across a variety of data sources.
.NET Framework

• C# programs run on the .NET Framework.
  – The .NET Framework includes a virtual execution system known as the CLR - common language runtime and a unified set of class libraries.
  – The CLR is the commercial implementation by Microsoft of the CLI - common language infrastructure.
    • It serves as the basis for creating execution and development environments in which languages and libraries work together seamlessly.
  – Source code written in C# is compiled into an IL - intermediate language that conforms to the CLI specification.
    • The IL code and resources, such as bitmaps and strings, are stored on disk in an executable file called an assembly.
    • The executable file typically will have an extension of .exe or .dll.

Relationships - C# Source Code and .NET Framework
C# and .NET

- The relationship between C# and the .NET Framework is unique.
- It is somewhat similar to the relationship between Java and the Java Virtual Machine.
  - However there are several major differences.
    - C# is not the only language that can be used to write .NET Framework applications.
      - .NET Framework applications are known as managed applications.
    - .NET or managed applications run in native machine-language and are not interpreted.
    - C# or managed applications do not run in a sandbox.

.NET Blueprint - Microsoft's Vision
MSIL and Common Type System

- All .NET languages produce "virtual" machine language known as MSIL or Microsoft Intermediate Language.
  - MSIL is similar to Java bytecode in that it targets the Java Virtual Machine.
  - MSIL is converted to machine code when loaded.
- All .NET languages use a CTS - Common Type System.
  - Language-specific types are mapped to .NET types.
    - Value types: primitives and structs (records)
    - Reference types: objects, pointers, interfaces
  - This is how cross-language interoperability is achieved.

.NET - from Our Perspective

- An execution environment called the CLR:
  - Converts MSIL to machine code.
  - Is responsible for security.
- It is a class library:
  - Common to VB, C# "managed" C++, etc.
- It provides translators or languages.
What are .NET Programs?

- .NET programs can be written in:
  - C#, VB.NET, and Managed C++.
- All .NET languages use the same class library.
  - Do not use standard VB or C++ library.
  - The .NET Framework Class Library replaces the built-in library.
    - The .NET Foundation Class Library, known as the FCL, provides capabilities and features similar to Java Class Libraries.

Framework Class Libraries
How Translation Works in .NET

- VB
- C#
- Script
- ...

Install time Code Gen
Assembly: MSIL + Metadata
Development

Common Language Runtime
JIT Code Gen
Native Code
Deployment

Result is Managed by CLR

- CLR - Common Language Runtime is the heart of .NET.
  - ALL .NET code is managed.
  - .NET cannot run unmanaged code or use a legacy library.
    - Example: cout << "Hi"
      - Can not be written.
    - However, it is currently possible to run both .NET and unmanaged code

- What does CLR do?
  - Provides compilation and class loading.
  - Handles security, range-checking, and exceptions.
  - Provides garbage collection.
  - Provides thread support, class library interfaces, and COM support.
Parts of the CLR

- Base Class Library Support
- Thread Support
- COM Marshaler
- Type Checker
- Exception Manager
- Security Engine
- Debug Engine
- IL to Native Compilers
- Code Manager
- Garbage Collector
- Class Loader

Lesson 2

Compiling and Running
Objectives

You will learn:

• How to compile and run a program using the command line.
• How to compile and run a program using the IDE Visual Studio.
• How to create a simple console, window, and component application.

Compile and Run the Program

• In order to compile the main program and use the library, run the following:
  csc /r:ex04.dll testmain.cs
• In order to run the program enter:
  testmain.
  – The /r switch in compiling the component indicates to the C# compiler that the source code in testmain.cs references objects implemented in ex04.dll.
Compile and Run the Program

• It is important to recognize that:
  – The referenced file is the binary assembly file name1.dll, not the source code file name2.cs.
  – The omission of the /Target switch indicates to the compiler that FibTest.cs should be compiled into the default assembly type.
  - This is a console application.

Visual Studio and Windows

• Microsoft Visual Studio offers several ways to develop Windows-based applications.
• Visual Studio can be used for creating Windows-based applications and UI - user interfaces by:
  – using Windows Forms.
  – using either Visual Studio or the .NET Framework Software Development Kit.
• Windows Win32-based applications can be created by using the Visual Studio Project Wizard.
• Starting with Visual Studio 2008, Windows-based applications can be created by using WPF - Windows Presentation Foundation.
Windows-based Application Development Tools

Windows-based application development tools include:
• Visual designers for Windows Forms with drag-and-drop controls.
• Visual designers for Windows Presentation Foundation.
• Code-aware editors that include statement completion, syntax checking, and other IntelliSense features.
• Integrated compilers and debugger.
• Project management tools for creating, managing and deploying application files
  – The deployment can be local, over an intranet or the Internet.

Create a Windows Application Project

• Perform the following steps in order to create a windows application project.
  – On the File menu, point to New, and then select Project.
    • In the Project Types pane, choose the programming language to be used.
    • In the Templates pane, choose Windows Application for Visual C# projects.
    • In the Name text box, specify a name for the project which is unique to the application's functionality.
    • In the Location text box, enter the directory in which the project is to be saved or click the Browse button to navigate to it.
Create a Windows Application Project

Windows Project Development
Simple Windows Application

• Managed applications are applications that run on the .NET Framework.
  – The following is an example of a simple C# or managed application.

```csharp
using System.Windows.Forms;
using System.Drawing;

class MyForm : Form{
    public static void Main(){
        Application.Run(new MyForm());
    }

    protected override void OnPaint(PaintEventArgs e){
        e.Graphics.DrawString("Hello SysEd!",
            new Font("Arial", 35),
            Brushes.Blue, 10, 100);
    }
}
```
Simple Windows Application

- The source code displays the text "Hello SysEd!" in a window.
  - C# has a C-based syntax, but with objects like C++ or Java.
- Every function in C# is a method of a type.
- The MyForm class is defined to derive its functionality from the Form class which is part of the .NET Framework Class Library.
  - In addition it defines two new methods, Main() and OnPaint().

Simple Windows Application

- All C# or .NET applications must have a static method named Main() defined to be the entry point of the application.
  - Methods declared as static do not require an object instance to be called.
  - They are similar to global functions and are often referred to as type-methods rather than instance methods.
- The OnPaint() method is an override of a virtual method on the Form class.
  - It is called when the window needs to paint itself.
    - The sample will use this method to draw the text.
Compiling and Running the Program

• Once the .NET Framework has been installed, the Framework SDK will also be available.
  – It provides the command-line compiler for C#.
• The C# compiler is called csc.exe and exists in the directory under C:\WINDOWS.
  – Use the command dir csc.exe /s to locate the directory and store the directory name in the path.
  – A common directory would be: C:\WINDOWS\Microsoft.NET\Framework\v3.5.

Compiling and Running the Program

• Assuming the file name of the program is ex01.cs, compile the program using:
  csc /Target:winexe ex01.cs
• In order to run the program, enter ex01 at the command prompt.
• The Visual Studio is typically used for large projects; the command line compiler is used for small tests and scripts.
## Console Application

```csharp
using System;
// Console application
// How many rabbits you get after a certain number of generations
class App{
    public static void Main(String[] args) {
        try {
            Int32 iterations = Convert.ToInt32(args[0]);
            if (iterations > 138) {
                throw new Exception();
            }
            Decimal lastNum = 1;
            Decimal secondToLastNum = 0;
            while (iterations-- > 0) {
                Decimal newNum = lastNum + secondToLastNum;
                Console.WriteLine(newNum);
                secondToLastNum = lastNum;
                lastNum = newNum;
            }
        } catch {
            Console.WriteLine(
                "Usage: Rabbits [Fib Index]\n"+
                "\t[Fib Index] < 139\n";
            )
        }
    }
}
```

## Compile and Run Program

- This is an example of a console application.
- Once this application is built, it can be run from the command line, and passed as an argument indicating the number of generations that will be calculated.
- Use the following command to build this file using the command line compiler

```bash
csc ex02.cs
```
- It will not be necessary to use the /Target switch.
Console Application Description

• ex02.cs defines a class (arbitrarily named App) which defines an entry point function named Main().
• The Main() method does something a little different by taking a parameter defined as an array of String objects.
  – These are the command line arguments passed to the program.
• The program uses structured exception handling to handle errors.
• C# uses C syntax for its loop constructs.
  – The while loop syntax in Rabbits.cs is identical to what it would be in C, C++, or Java.

Console Application Description

• The static WriteLine() method of the Console type defined in the Framework Class Library is used to output text to the console window.
  – An instance of the Console type was not necessary to call the method.
  – This is because WriteLine() is defined as a static method.
• The program uses a numeric type called Decimal.
**Window Application with Controls**

```csharp
using System;
using System.Drawing;
using System.Windows.Forms;

class App{
    public static void Main(){
        Application.Run(new TribbleForm());
    }
}

class TribbleForm:Form{
    TextBox generationsTextBox;
    ListBox fibList;
    public TribbleForm() {
        generationsTextBox = new TextBox();
        generationsTextBox.Location = new Point(16, 16);
        tribbleButton = new Button();
        tribbleButton.Location = new Point(16, 48);
        tribbleButton.Size = new Size(100, 20);
        tribbleButton.Text = "Tribble Count";
        AcceptButton = tribbleButton;
        fibList = new ListBox();
        fibList.Location = new Point(16, 88);
        fibList.Size = new Size(192, 134);
        fibList.Anchor = AnchorStyles.Top | AnchorStyles.Bottom;
        Controls.AddRange(new Control[]{generationsTextBox, tribbleButton, fibList});
        Text = "Tribble Calculator";
    }
    
    void OnClick(Object sender, EventArgs e){
        try{
            Int32 iterations = Convert.ToInt32(generationsTextBox.Text);
            if(iterations > 138)
                throw new Exception();
            fibList.Items.Clear();
            Decimal lastNum = 1;
            Decimal secondToLastNum = 0;
            while(iterations-- > 0){
                Decimal newNum = lastNum + secondToLastNum;
                fibList.Items.Add(newNum);
                secondToLastNum = lastNum;
                lastNum = newNum;
            }
            fibList.SelectedIndex = fibList.Items.Count - 1;
        }catch{
            MessageBox.Show("Enter a number from 1-138");
        }
    }
}
```

**Event Processing**

```csharp
void OnClick(Object sender, EventArgs e){
    try{
        Int32 iterations = Convert.ToInt32(generationsTextBox.Text);
        if(iterations > 138)
            throw new Exception();
        fibList.Items.Clear();
        Decimal lastNum = 1;
        Decimal secondToLastNum = 0;
        while(iterations-- > 0){
            Decimal newNum = lastNum + secondToLastNum;
            fibList.Items.Add(newNum);
            secondToLastNum = lastNum;
            lastNum = newNum;
        }
        fibList.SelectedIndex = fibList.Items.Count - 1;
    }catch{
        MessageBox.Show("Enter a number from 1-138");
    }
}
```
GUI Classes

- Tribbles.cs defines two classes.
  - An App class, which has a static Main() method to be used as the program’s entry point.
  - The TribbleForm class, which is derived from form and implements the program’s window.

- Most of the programs functionality exists in the TribbleForm class’ constructor and the OnClick() handler method for the button.

- Although the GUI code can be written, it is far more common to use a forms designer.

GUI Classes

- The .NET Framework ships with a free forms designer called WinDes.exe.

- Visual Studio.NET also provides forms designers.
  - Both allow a GUI to be designed using graphical tools; the designer will then create the code.

- C# GUI programs use a set of classes in the FCL known as the Windows Forms classes.
Creating Code
Library Assemblies

• It is important to know how to create executables with C#.
• As software evolves, component development will continue to become increasingly important.
  – Increasingly, application code is going to be housed in reusable objects.
  – These reusable types will often exist in binary modules or assemblies external to the main executable.

Component Program

using System;
public class Fib{
    Decimal current;
    Decimal last;
    public Fib() {
        current = 1;
        last = 0;
    }
    private Fib(Decimal last, Decimal secondToLast) {
        current = last+secondToLast;
        this.last = last;
    }
    public Fib GetNext() {
        return new Fib(current, last);
    }
    public Decimal Value {
        get{return current;}
    }
}
Compiling the Library

- In order to compile this object into a library using the command line compiler, perform the following:

  csc /Target:library ex04.cs
  - The /Target switch is used to indicate that the assembly being built is a library.
  - This will create an assembly named ex04.dll.
    - ex04.dll does not have a static Main() entry method defined, and it cannot be executed directly.
    - If a non-library assembly is built without an entry point method defined, the compiler will generate an error message.

Test Library Routine

```csharp
using System;

class App{
    public static void Main(){
        Int32 index = 50;
        Fib obj = new Fib();
        do{
            Console.WriteLine(obj.Value);
            obj = obj.GetNext();
        }while(index-- != 0);
    }
}
```