Review of Visual C# Syntax

Module Overview

- Overview of Writing Application by Using Visual C#
- Data Types, Operators, and Expressions
- Visual C# Programming Language Constructs

Lesson 1: Overview of Writing Application by Using Visual C#

- What Is the .NET Framework?
- Key Features of Visual Studio
- Templates in Visual Studio
- Creating a .NET Framework Application
- Overview of XAML

What Is the .NET Framework?

- CLR
  - Robust and secure environment for your managed code
  - Memory management
  - Multithreading
- Class library
  - Foundation of common functionality
  - Extensible
- Development frameworks
  - WPF
  - Windows store
  - ASP.NET
  - WCF

Key Features of Visual Studio

- Intuitive IDE
- Rapid application development
- Server and data access
- IIS Express
- Debugging features
- Error handling
- Help and documentation

Templates in Visual Studio

- Console Application
- Windows Forms Application
- WPF Application
- Windows Store
- Class Library
- ASP.NET Web Application
- ASP.NET MVC 4 Application
- WCF Service Application

Overview of XAML

- XML-based language for declaring UIs
- Uses elements to define controls
- Uses attributes to define properties of controls

```
<Label Content="Name:" HorizontalAlignment="Left"
Margin="72,43,0,0" VerticalAlignment="Top" />

<TextBox HorizontalAlignment="Left" Height="23" Margin="141,43,0,0"
Text="" VerticalAlignment="Top" Width="120" />

<Button Content="Click Me!" HorizontalAlignment="Left"
Margin="119,84,0,0" VerticalAlignment="Top" Width="75" />
```

Lesson 2: Data Types, Operators, and Expressions

- What are Data Types?
- Expressions and Operators in Visual C#
- Declaring and Assigning Variables
- Accessing Type Members
- Casting Between Data Types
- Manipulating Strings

What are Data Types?

- int – whole numbers
- long – whole numbers (bigger range)
- float – floating-point numbers
- double - double precision
- decimal - monetary values
- char - single character
- bool - Boolean
- DateTime - moments in time
- string - sequence of characters

Expressions and Operators in Visual C#

Example expressions:

- + operator

```
a + 1
```

- / operator

```
5 / 2
```

- + and – operators

```
a + b - 2
```

- + operator (string concatenation)

```
"ApplicationName: " + appName.ToString()
```

Declaring and Assigning Variables

- Declaring variables:

```
int price;
// OR
int price, tax;
```

- Assigning variables:

```
price = 10;
// OR
int price = 10;
```

- Implicitly typed variables:

```
var price = 20;
```

- Instantiating object variables by using the new operator

```
ServiceConfiguration config = new ServiceConfiguration();
```

Accessing Type Members

- Invoke instance members

```
<instanceName>.<memberName>
```

- Example:

```
var config = new ServiceConfiguration();

// Invoke the LoadConfiguration method.
config.LoadConfiguration();

// Get the value from the ApplicationName property.
var applicationName = config.ApplicationName;

// Set the .DatabaseServerName property.
config.DatabaseServerName = "78.45.81.23";

// Invoke the SaveConfiguration method.
config.SaveConfiguration();
```

Casting Between Data Types

- Implicit conversion:

```
int a = 4;
long b = 5;
b = a;
```

- Explicit conversion:

```
int a = (int) b;
```

- System.Convert conversion:

```
string possibleInt = "1234";
int count = Convert.ToInt32(possibleInt);
```

Manipulating Strings

- Concatenating strings

```
StringBuilder address = new StringBuilder();
address.Append("23");
address.Append(", Main Street");
address.Append(", Buffalo");
string concatenatedAddress = address.ToString();
```

- Validating strings

```
var textToTest = "hell0 w0rld";
var regularExpression = "\\d";

var result = Regex.IsMatch(textToTest, regularExpression,
RegexOptions.None);

if (result)
{
   // Text matched expression.
}
```

Lesson 3: Visual C# Programming Language Constructs

- Implementing Conditional Logic
- Implementing Iteration Logic
- Creating and Using Arrays
- Referencing Namespaces
- Using Breakpoints in Visual Studio 2013
-

Implementing Conditional Logic

- if statements

```
if (response == "connection_failed") {. . .}
else if (response == "connection_error") {. . .}
else { }
```

- select statements

```
switch (response)
{
   case "connection_failed":
      . . .
      break;
   case "connection_success":
      . . .
      break;
   default:
      . . .
      break;
}
```

Implementing Iteration Logic

- for loop

```
for (int i = 0 ; i < 10; i++) { ... }
```

- foreach loop

```
string[] names = new string[10];
foreach (string name in names) { ... }
```

- while loop

```
bool dataToEnter = CheckIfUserWantsToEnterData();
while (dataToEnter)
{
    ...
    dataToEnter = CheckIfUserHasMoreData();
}
```

- do loop

```
do
{
    ...
    moreDataToEnter = CheckIfUserHasMoreData();
} while (moreDataToEnter);
```

Creating and Using Arrays

- C# supports:
  - Single-dimensional arrays
  - Multidimensional arrays
  - Jagged arrays

- Creating an array

```
int[] arrayName = new int[10];
```

- Accessing data in an array:
  - By index

```
int result = arrayName[2];
```

  - In a loop

```
for (int i = 0; i < arrayName.Length; i++)
{
    int result = arrayName[i];
}
```

Referencing Namespaces

- Use namespaces to organize classes into a logically related hierarchy
- .NET Class Library includes:
  - System.Windows
  - System.Data
  - System.Web
- Define your own namespaces:

```
namespace FourthCoffee.Console
{
   class Program {. . .}
```

- Use namespaces:
  - Add reference to containing library
  - Add using directive to code file

Using Breakpoints in Visual Studio

- Breakpoints enable you to view and modify the contents of variables:
  - Immediate Window
  - Autos, Locals, and Watch panes
- Debug menu and toolbar functions enable you to:
  - Start and stop debugging
  - Enter break mode
  - Restart the application
  - Step through code

Text Continuation

Lab: Developing the Conditionals

- Exercise 1: Implement an array and calculate the average of the elements.
- Exercise 2: Implement a program for calculate the number of the balotto lottery.
- Exercise 3: Write a program that evaluates two variables: one true single condition, two true conditions and a nested condition.
- Exercise 4: Implement: [for] statement counting up by 1, down by 2, by multiples of 5, a [foreach] of numbers, a [foreach] of strings, a while and a do… while.

Estimated Time: 60 minutes

Text Continuation

Module Review and Takeaways

- Review Question(s)

Text Continuation