

## C# Programming: Hands-On - 4 Days

### *Course 419 Overview*

- You Will Learn How To**
- Create, compile and run object-oriented C# programs using Visual Studio
  - Write and understand C# language constructs, syntax and semantics
  - Produce programs for the Web and desktop using solid multilayer architectures
  - Develop reusable .NET components via interface realization and standard design patterns
  - Leverage the major namespaces and classes of the .NET Framework
  - Access databases using Language Integrated Query (LINQ) and the Entity Framework (EF)

**Course Benefits** C# is a modern, object-oriented programming language intended to create simple yet robust programs. Designed specifically to take advantage of CLI features, C# is the core language of the Microsoft .NET framework. In this course, you gain the skills to exploit the capabilities of C# and of the .NET Framework to develop programs useful for a broad range of desktop and Web applications.

**Who Should Attend** Anyone interested in programming in C#. Experience with a modern language such as VB, Java, Pascal or C/C++ is assumed. Those with only COBOL, RPG, SQL, HTML or similar experience should consider taking Course 502, "Programming with .NET Introduction."

**Hands-On Training** You gain experience creating your own C# application. Hands-on exercises include:

- Writing and compiling C# programs using Visual Studio
- Building C# classes and inheritance hierarchies
- Writing desktop and Web applications with Windows Forms and Web Forms
- Constructing and deploying custom .NET components
- Implementing data-query logic for databases using LINQ and EF
- Accelerating development with the .NET Framework library

## C# Programming: Hands-On - 4 Days

### Course 419 Outline

#### Introduction to the C# Language

##### The evolution of C#

- Comparing different versions of C#
- Expressing C# models in UML

#### C# and the .NET infrastructure

- Common Language Infrastructure (CLI)
- Managed code philosophy
- Common Intermediate Language (CIL) and metadata

#### Language Fundamentals

##### Data types and control constructs

- Declaring implicit and explicit variables
- Value and reference types
- Unicode characters and strings

##### Defining and calling methods

- The Main method specification
- Passing arguments and returning values
- The scope and lifetime of variables
- Named and symbolic methods
- Handling exceptions
- Recovering resources

##### Employing .NET library classes

- Avoiding collisions by using namespaces
- Performing I/O using the stream class and serialization
- Standard and Generic Collections

#### Developing C# Classes

##### Defining classes

- Encapsulating attributes with methods and properties
- Providing consistent initialization using constructors
- Overloading methods and constructors
- Achieving reuse through inheritance and polymorphism

##### Creating and using objects

- Allocating object memory with `new`
- Passing initial values to constructors
- Choosing value or reference allocation
- Boxing and unboxing
- Invoking methods and accessing properties

#### Interconnecting Objects

##### Associating classes

- Manipulating references
- Physical vs. logical equivalence
- Selecting collection library classes

- Increasing reliability using generics

##### Exposing interfaces

- Defining an interface specification
- Implementing an interface in a class
- Interface polymorphism
- Events and delegates

#### Simplifying Component Development

##### Component features of .NET

- Manifests and assemblies
- Deploying components and applications
- .NET assembly metamodel

##### Writing .NET components in C#

- Creating and calling custom components
- Extending `System.ComponentModel.Component`

##### Interfacing legacy components

- Accessing COM/DCOM
- Tools for forward and backward compatibility
- Calling existing components

##### Integrating C# with other languages

- Harmonizing components through the CLI
- Accessing metadata
- Handling cross-language differences

#### Implementing and Enhancing C# Solutions

##### Building multitier applications

- Leveraging solid architectural patterns (MVC and EDM)
- Substituting the user interface
- Coding industry-standard design patterns in C#
- Distributing a C# application

##### Working with relational databases

- Accessing databases with the Entity Framework (EF)
- Integrating C# extended features with LINQ

##### Standards and versions

- Standardization via ECMA/ISO
- Features in various C# standards

##### Advanced techniques

- Automating documentation with XML

- Implementing the `IEnumerable<T>` interface
- Invoking extension methods
- Employing events, delegates and lambda expressions
- Specifying development attributes