



transforming performance through learning

## **Introduction to the Microsoft C# 4.0 Programming Language - QACSHPLI40**

### **Course Details**

Days 3  
Course code QACSHPLI40

### **Course Outline**

#### **Overview**

Microsoft's .NET Framework presents developers with unprecedented opportunities. From web applications to desktop and mobile platform applications - all can be built with equal ease, using substantially the same skill-set. But to make the most of this potential, developers must have a thorough grasp of core language skills and OO programming concepts.

This course is intended for developers who will use C# to write .NET Framework applications and who are new to the C# programming language. This includes those currently working with Visual Basic 6.0, C++ or Java.

This three-day workshop concentrates on the C# programming language itself, to prepare delegates fully in readiness for exploring the .NET Framework. From basic procedural syntax to sophisticated object-oriented programming techniques, delegates will learn how to write .NET applications with code that is robust and maintainable.

The course is presented as a mixture of lectures and hands-on exercises. Practical sessions follow all main topics, designed to reinforce the points covered. Additional information is provided in appendices to extend the learning experience after the course has been completed.

#### **Prerequisites**

- Delegates must understand the fundamentals of programming and should have some existing knowledge of object oriented programming concepts.
- This is a course for developers. Good keyboard skills are an absolute necessity.

#### **On completion**

At the end of this course, you will be able to:

- Write efficient procedural code that includes sequence, selection and iteration constructs
- Create and use classes and structures (types), including fields, properties and methods
- Use private, internal, protected and public visibility modifiers
- Create derived classes that inherit from custom-written or .NET Framework classes
- Create interfaces and apply techniques of polymorphism effectively and appropriately
- Build exception-handling into methods, to create robust, user-friendly applications
- Work effectively with delegates and events and understand how they operate
- Leverage the power of C# features such as indexers and iterators
- Work with generic types and efficiently manage resources
- Version assemblies and know how .NET searches and loads the correct DLL's

## **Course Outline**

### **Module 1: Introduction to .NET & C#**

- The .NET Framework; The Common Language Runtime; The Common Type System
- C# Features; Introduction to namespaces and assemblies

### **Module 2: Language Fundamentals**

- Procedures and statements; Data types; Declaring variables; Assignments
- Conversion; Arithmetic and other operators
- Control constructs; by value, by reference, named and optional parameters

### **Module 3: Types I**

- Type concepts; Classes; Reference types
- Fields, properties and methods; C#3 Auto-implemented properties
- Accessibility modifiers; Construction and chaining
- Instance members; Keyword 'this'
- The 'null' reference

## **Module 4: Types II**

- Structs; Value types
- Object Initialisers
- Static; Const & ReadOnly
- The Singleton & Factory patterns
- Partial classes
- Enumerated types

## **Module 5: Exception Handling**

- Errors vs. Exceptions; The 'try' block; The 'catch' block; The 'finally' block; Using 'throw'
- Creating your own exceptions; 'checked' and 'unchecked' expressions.

## **Module 6: Inheritance & Polymorphism**

- Concept of inheritance; Substitutability; Extending a simple class
- 'virtual', 'override' and 'sealed' modifiers
- Polymorphism
- Upcasting and safe downcasting

## **Module 7: Abstract Classes & Interfaces**

- Abstract classes; Abstract methods and properties
- Polymorphism with interfaces; Multiple interfaces

## **Module 8: Generics & Collections**

- Arrays vs Collections; Array syntax
- Generic concepts; Using Generic collection classes; `List<T>`;
- Generic interfaces; `IComparable<T>`, `IComparer<T>`; & sorting
- Indexers; C# Iterators, `IEnumerable<T>`, `IEnumerator<T>`;
- Using generic methods
- Co & Contra-variance

- Constraints; Nullable types
- Boxing / UnBoxing issues

#### **Module 9: Delegates & Events**

- Delegates explained; Working with delegates; Creating your own delegate types; Events; Evolution of syntax for creating delegate instances; Generic delegates

#### **Module 10: Managing Resources**

- Garbage collection and its impacts; Finalizers; The 'Dispose' pattern; IDisposable
- The using statement for deterministic resource management.

#### **Module 11: The Way Ahead**

- Review
- Follow-on courses

#### **Appendix Module: Namespaces & Assemblies**

- Namespaces; The 'using' statement for namespaces; Assemblies
- DLLs at compile time and run Time; The Global Assembly Cache (GAC)
- Versioning using public/private key cryptography

#### **Appendix Module: Operator Overloading**

- Why operator overloading is useful; When to overload
- Implicit conversions; explicit conversions

#### **Appendix Module: C# & .NET timeline**

- Which versions/functionality came when; C# syntax changes by version

#### **Appendix Module: Working with Text**

- Class String; Class StringBuilder; Formatting Strings; Regular Expressions

QA reserves the right to improve the specification and format of its courses for the benefit of its customers without notice to the customer.

