

# **.NET Framework Fundamentals**

(version 1.0)

## **I. Overview**

The course objective is to give the students a stable basis of knowledge and skills for working as a software engineer for the Microsoft .NET platform. It covers the fundamental set of knowledge and skills that any .NET developer should have: C# language, object-oriented programming, standard .NET Framework APIs, and Windows Forms GUI applications. The course is based on .NET Framework 3.5, MS Visual Studio 2008 and MS SQL Server 2008.

## **II. Requirements**

Elementary computer skills: required.

Computer English: initial level.

Preliminary programming skills: required (Introduction to Programming with C#).

## **III. Course Program**

### **1. Course Overview (lectures: 1, exercises: 0, homework: 0)**

- Course scope, curriculum, homework, lectures, and exercises
- Introduction to Team Foundation Server (TFS)

### **2. .NET Framework Architecture (lectures: 2, exercises: 0, homework: 1)**

- Common Language Runtime (CLR)
- Intermediate language (MSIL)
- Assemblies and .NET Framework execution model
- Creating, compiling and running .NET applications

### **3. The C# Programming Language – Part 1 (lectures: 1, exercises: 1, homework: 0)**

- C# language design principles
- C# programs structure
- Identifiers; Keywords
- Primitive data types; Enumerations; Typecasting and conversion
- Variables; Declarations

#### **4. The C# Programming Language – Part 2 (lectures: 2, exercises: 1, homework: 1)**

- Operators
- Expressions; Statements
- Control structures, conditionals, loops (if, switch, for, while, do ... while)
- Console input/output

#### **5. Object-Oriented Programming Concepts (lectures: 2, exercises: 0, homework: 2)**

- Object-oriented programming and design fundamentals
- Classes, objects and interfaces
- Inheritance, abstraction, encapsulation and polymorphism

#### **6. Classes and Objects (lectures: 1, exercises: 1, homework: 2)**

- Creating and using objects
- Defining classes and class members (fields, properties, constants)
- Defining and using methods; Method overloading
- Defining and using properties
- Constructors
- Static members and static constructor
- Structures

#### **7. Inheritance and Polymorphism (lectures: 1, exercises: 1, homework: 3)**

- Inheritance and polymorphism
- Abstract classes
- Namespaces

#### **8. Interfaces, Indexers and Operators (lectures: 1, exercises: 1, homework: 3)**

- Defining, implementing and using interfaces
- Defining and using indexers
- Overloading operators

#### **9. Generics (lectures: 1, exercises: 1, homework: 3)**

- Generic methods and generic classes

## **10. The C# Programming Language – Part 3 (lectures: 2, exercises: 1, homework: 3)**

- New enhancements in C# 3.0
- Automatic properties
- Object and collections initializers
- Implicit typed local variables (var keyword)
- Lambda expressions
- Extension methods
- Anonymous types

## **11. Common Type System (CTS) and System.Object (lectures: 1, exercises: 1, homework: 4)**

- Built-in types hierarchy
- The base type System.Object; Overriding equality and hash code methods
- Object cloning and ICloneable
- Implementing IComparable

## **12. Value Types and Reference Types (lectures: 1, exercises: 1, homework: 1)**

- Value types and reference types; Representation in the memory
- Boxing and unboxing value types
- Passing arguments by value, reference and as output (in, out and ref parameters)
- Nullable types

## **13. Exceptions Handling (lectures: 1, exercises: 1, homework: 2)**

- Error handling and exceptions
- Catching and throwing exceptions
- Defining and using custom exception classes

## **14. High-Quality Code Construction (lectures: 2, exercises: 0, homework: 0)**

- What is high-quality programming code?
- High-quality software design (cohesion, coupling, complexity, reusability, fan-in, fan-out, etc.)
- High-quality methods (cohesion, coupling, naming, parameters, length, etc.)
- Defensive programming and exceptions
- Best practices using variables (naming, scope, span, lifetime, etc.)
- Naming variables, methods, classes and other identifiers

- Best practices using conditionals and loops
- Self-documenting code and effective comments

## **15. Working with Strings (lectures: 1, exercises: 1, homework: 4)**

- The Unicode standard
- Characters, strings and string processing: creating, concatenating, extracting substrings, searching, comparing, and splitting
- Building and modifying strings with StringBuilder

## **16. String Formatting, Cultures and Internationalization (lectures: 2, exercises: 2, homework: 4)**

- String, numbers and date formatting
- Cultures
- Parsing numbers and dates
- Encodings and conversions

## **17. Regular Expressions (lectures: 2, exercises: 2, homework: 4)**

- The regular expressions language – literals and meta-characters, character classes, quantifiers, grouping characters, etc.
- Regular expressions API in .NET Framework
- Searching, extracting, validating, splitting and replacing text by regular expressions

## **18. Delegates and Events (lectures: 2, exercises: 1, homework: 3)**

- Defining and using delegates and events
- Anonymous delegates
- Predicate methods
- Generic EventHandler delegate

## **19. Algorithms and Complexity (lectures: 1, exercises: 0, homework: 2)**

- Algorithms complexity
- Asymptotic notation
- Analyzing and computing complexity

## **20. Collections Classes (lectures: 2, exercises: 2, homework: 4)**

- Lists and representation – ArrayList, List<T>
- Queues and representation – Queue, Queue<T>

- Stacks and representation – Stack, Stack<T>
- Hash-tables and representation – HashTable, Dictionary<TKey, TValue>
- Sets and HashSet<T>
- Collections Interfaces: IEnumerable<T>, ICollection<T>, IList<T>, IDictionary<K, T>
- Iterators: IEnumerable<T> and IEnumerator<T>

## **21. Graphs and Traversal Algorithms (lectures: 1, exercises: 1, homework: 3)**

- Graphs – basic concepts and representation
- Depth First Search (DFS) algorithm
- Breath First Search (BFS) algorithm

## **22. Memory and Resource Management (lectures: 1, exercises: 0, homework: 2)**

- Memory management, managed heap and garbage collection
- Resource management, IDisposable and finalizers

## **23. Input/Output (lectures: 2, exercises: 2, homework: 4)**

- Binary streams and text streams
- File streams
- Files and directories

## **24. Attributes (lectures: 1, exercises: 0, homework: 2)**

- Using attributes, attribute parameters, attribute targets
- Defining custom attributes

## **25. Reflection (lectures: 1, exercises: 1, homework: 2)**

- Loading assemblies and classes
- Exploring metadata – assemblies, classes and class members
- Dynamically instantiating classes and invoking methods

## **26. Serialization (lectures: 1, exercises: 1, homework: 1)**

- Automatic and custom serialization; Formatters
- XML and binary serialization

## **27. Introduction to Windows Forms (lectures: 1, exercises: 1, homework: 3)**

- Windows Forms programming model
- Windows Forms basic classes: Component, Control, ScrollableControl, ContainerControl
- Windows Forms events model, event queue and controls rendering model

- Basic controls: Form, Label, TextBox, Button
- Handling events

## **28. Windows Forms Dialogs and Advanced Controls (lectures: 1, exercises: 3, homework: 3)**

- Creating and using dialog boxes
- Advanced controls (panels, pictures, list boxes, combo boxes, menus, status bars, toolbars, file dialog, etc.)
- Data validation

## **29. Windows Forms Data Binding and Data Bound Controls (lectures: 2, exercises: 2, homework: 6)**

- Data binding, navigation and binding sources
- Data bound controls (DataGridView, ListBox, ComboBox, etc.)

## **30. Windows Forms Advanced Topics (lectures: 1, exercises: 1, homework: 4)**

- Forms inheritance
- Graphics primitives and System.Drawing
- Custom controls
- Drag and drop

## **31. Threads and Synchronization (lectures: 2, exercises: 1, homework: 6)**

- Multithreading overview – processes and threads
- Using the Thread class – creating, starting and stopping threads; Threads lifecycle
- Threads synchronization – race conditions, critical sections, monitors

## **32. Asynchronous Execution (lectures: 1, exercises: 2, homework: 4)**

- Using the built-in thread pool (the ThreadPool class)
- Asynchronous execution in Windows Forms
- Asynchronous execution with BeginInvoke() / EndInvoke()

## **33. Networking Fundamentals (lectures: 2, exercises: 0, homework: 0)**

- Basic Internet concepts: OSI seven-layer model, TCP/IP protocol suite, protocols, services, IP address, network interface, DNS, TCP, UDP, sockets, etc.

## **34. Network Programming (lectures: 2, exercises: 2, homework: 12)**

- TCP and UDP sockets
- Accessing Internet resources through URL
- E-mail API

### **IV. Training Duration**

Lectures: 48 hours

Exercises: 35 hours

Homework: 98 hours

Allocation: ~ 14 weeks, 2 times \* 3 hours at week