Course Description - C# Programming

LEANID VAITSEKHOVICH

BrSTU, 2012

Table of content

1. INTRODUCTION

The document provides a description of a training course on C# programming.

2. OVERVIEW

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, students 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.

3. COURSE OBJECTIVES

Students will learn how to

- Create, compile and run object-oriented C# programs using Visual Studio
- Write and understand C# language constructs, syntax and semantics
- 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)

4. COURSE DURATION

68 hours:
Lectures – 36 hours
Practical classes – 16 hours
Independent work – 16 hours

5. PREREQUISITES

Experience with a modern language such as VB, Java, Pascal or C/C++ is assumed. Knowledge of object oriented programming is needed as well. The course exceeds the basic level and can be considered as intermediate.

6. HANDS-ON TRAINING

Students gain experience creating their own C# application. Hands-on exercises include:

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

7. COURSE CONTENT

Lecture 1. Introducing .NET and C#.

What exactly is .NET? Multiple .NET programming languages and VS.NET. Intermediate language. The .NET Common Language Runtime. Competing in parallel with Java technologies. Common language infrastructure. Other .NET-related technologies. Introducing C#. Potent combo of Java and C++. The simplest program “Hello World!!!”. How to compile and run the code examples. Disassembling an assembly file.

Lecture 2. C# Types and Operators.

C# type categorization. Value types. Reference types. Casting for value/reference types. Boxing/Unboxing. Implicitly Typed Variables. Common typing with other .NET languages. Operators and their precedence in C#. typeof operator. Comparing Object.GetType, Type.GetType and typeof. The == operator. The is/as operators. Using checked and unchecked.

Lecture 3. Method Issues.

Method modifiers. Method structure. Method parameters. Constructors and Destructors. Instance and Static Constructors. Constructor initializers and constructor chaining. Object initializers. Optional arguments. Named arguments. Method overloading. Variable numbers of parameters in C# methods. Abstract methods. Method overriding. Method overriding vs. Method hiding. Using keyword base. Static methods. Sealed methods.

Lecture 4. Class and Interface Issues.

Class modifiers. Class members. Creating an object with the new operator. Looking at System.Object. Class inheritance. Implementing interfaces. Interface name conflict resolution. Name hiding of interface methods. Sealed classes. Abstract classes. Static classes. Nested classes. Anonymous Types.

Lecture 5. Arrays. Strings. Regex.

The different types of arrays in C#: one-dimensional arrays, multi-dimensional arrays (rectangular arrays, jagged arrays). Using the System.Array class. The foreach loop. Introduction to C# Strings. String fields and properties. Instantiating a string object. String manipulation in C#. String instance methods. StringBuilder class in C#. String performance optimization. Learn Regular Expression (Regex).

Lecture 6. Miscellaneous Issues.

Access modifiers. Partial classes. Partial methods. C# constants and read-only fields. The using statement. Nullable types. The ?? Operator. Nullable objects and the relational and logical operators.

Lecture 7. C# Properties and Indexes. Operator Overloading. User-Defined Conversions/Casts.

Properties as accessor and mutator methods. Inheritance of properties. Auto-implemented properties. Use access modifiers with accessors. Property restrictions. Using indexers. Overloading indexers. Explaining operator overloading. User-defined conversions/casts. Explicit and implicit user-defined casts.

Lecture 8. C# Enums and Structures. C# Attributes.

Using enums. Specify the underlying type of an enumeration. Using structures. Differences between a struct and a class. Custom attributes: creating, attaching, obtaining. Named vs. positional parameters. Standard attributes: Obsolete, Conditional, AttributeUsage.

Lecture 9. Exception Handling.

Basic try-catch-finally. Nested try. User-Defined Exceptions. C# exception hierarchy. Examining System.Exception. Inner exceptions.

Lecture 10. C# Delegates and Events.

What are delegates? Declaration, instantiation and invocation. combining/removing delegates. Exception throwing in delegates. Anonymous methods. Lambda expressions. Generic event model. What are C# events?
Declaration of an event object. Invokation of the event. Providing and registering an event handler.

Lecture 11. Multi-Threading.

Important classes used for threading operations. Thread states and multi-threading in C#. Instantiating the thread class. Thread methods. Thread priorities. Foreground versus background threads. Using the lock keyword. Thread safety in .NET classes. Thread synchronization: Pulse and Wait methods.

Lecture 12. Reflection.

Classes which are significant in reflection. Retrieving the type of an instance. Retrieving the type from a name of a class. Retrieving methods from a type. Retrieving modules from an assembly. Dynamically invoking methods in late bound objects. Creating new types during runtime.

Lecture 13. File I/O.

Important classes used for I/O operations. Important exception classes for I/O operations. Copying, moving, and deleting files and directories. Reading from or writing to a binary file. Reading from and writing to text files.

Lecture 14. Add-on Serialization.

Understanding object serialization. The role of object graphs. Defining serializable types. Choosing a serialization formatter. Serializing objects using BinaryFormatter, SoapFormatter, XmlSerializer. Controlling the generated XML data. Customizing the Soap/Binary serialization process. A deeper look at object serialization. Customizing serialization using ISerializable and using attributes.

Lecture 15. C# Collection Classes.

Collections overview. The non-generic collections. The generic collections. The generic interfaces: IEnumerable, ICollection, IList, IDictionary. Implementing IComparable and IComparer. Accessing a collection via an Enumerator. Using iterators. Yield statement. Creating a named iterator. Collection initializers.

Lecture 16. ADO.NET.

Introduction. Data providers. ADO.NET objects. Working with disconnected data: DataSet and SqlDataAdapter. Adding parameters to commands. Using stored procedures.

Lecture 17. LINQ.

LINQ fundamentals. The general form of a query. How the data types in a query relate. Filter values, sort results, group results, joining. Query syntax vs. query methods. Extension methods. Create queries by using the query methods. Deferred vs. immediate query execution.

Lecture 18. Generics.

What are generics? The general form of a generic class. Constrained types. Creating a default value of a type parameter. Generic structures. Creating a generic method, generic delegate, generic interfaces. Generic class hierarchies. Overriding virtual methods in a generic class. Overloading methods.

8. METHOD OF EVALUATION

Evaluation Item	The Number of Times	Evaluation Proportion	Remarks
attendance		20%	80% of the classes
midterm exam			
final exam	1	10%	
final report			
test	4	20%	
presentation			
discussion			
homework		20%	
practice task	8	30%	All the practice tasks should be completed
etc			