# Learning to Program Using Visual Basic 2008



**Meet the expert:** Ken Getz is a featured instructor for several of our Visual Studio courses. He is a Visual Basic and Visual C# expert and has been recognized multiple times as a Microsoft MVP. Ken is a seasoned instructor, successful consultant, and the author or co-author of several best-selling books. He is a frequent speaker at technical conferences like Tech-Ed, VSLive, and DevConnections and he has written for several of the industry's most-respected publications including Visual Studio Magazine, CoDe Magazine, and MSDN Magazine.

Robert Green is a Visual Studio expert and a featured instructor for several of our Visual Basic and Visual C# courses. He is currently a Technical Evangelist in the Developer Platform and Evangelism (DPE) group at Microsoft. He has also worked for Microsoft on the Developer Tools marketing team and as Community Lead on the Visual Basic team. Robert has several years of consulting experience focused on developer training and is a frequent speaker at technology conferences including TechEd, VSLive, VSConnections, and Advisor Live.

**Prerequisites:** This course assumes that students have some programming background. No specific experience with Visual Studio 2008 or the .NET Framework is required. As with any such course, the more experience you bring to the course, the more you'll get out of it. This course moves quickly through a broad range of programming topics, but it does not require any prior .NET skills.

**Runtime:** 17:21:49

**Course description:** In this course, you will learn to use Visual Studio 2008 to explore the Visual Basic language. The course starts with a quick overview of the .NET platform, examining assemblies, Microsoft Intermediate Language, Visual Studio profiles, XML comments, IntelliSense, and debugging. From there, you will learn all the language features that you must internalize in order to create full-featured Web or Windows applications that make best use of the .NET platform. You will learn about data types, variables, and operators, along with all the important flow control structures. You will work through several examples demonstrating the power of the .NET Framework, and dig into creating and consuming your own classes and objects. The course moves on to working with data structures, such as arrays and collection classes, before finishing up with discussions of generics, handling exceptions and working with delegates and events. The course concludes by introducing the new LINQ-oriented features added to the .NET Framework 3.5, including anonymous types, lambda expressions, and more. By the end of this course, you will understand the important basic concepts that will allow you to start creating the applications you need.

**Course outline:**

**Introduction to .NET**
- Introduction
- Overview of .NET
- Why Use .NET
- Advantages of .NET
- Common Language Runtime
- Running Code and the CLR
- CLR and Compilers
- Just-in-Time Compiler (JIT)
- Framework Base Class Library
- Common BCL Namespaces
- .NET Languages
- Overview: Assemblies
- Overview: Manifest
- Create with .NET
- Use Command Line Compiler
- First VB App in Notepad
- Examine the Code in the App
- Manifest of the App
- Summary

**Introduction to Visual Studio**
- Introduction
- Working with Profile Settings
- Explore Profile Settings
- Set Save Project Location
- Create a New Project
- Windows in Visual Studio
- Pin / Unpin Windows
- Reset Window Layout
- Work with Files in the Project
- Set Project Properties
- Add a Reference
- View References
- View Object Browser
- Add Code to the Application
- Adding Imports statements
- Module Keyword Explained
- Adding Comments
- Using IntelliSense
- Build and Run the Application
- Summary

**Debugging**
- Introduction
- Overview
- Create a Sample
- Run the Sample
- Syntax Error
- View Error List Window
- Breakpoints
- Autos / Immediate Window
- Single Step Through Code
- Clear All Breakpoints
- Runtime Error
- Try/Catch Block
- Investigating Exceptions
- Logic Error
- Summary

**Intro to Variables / Data Types**
- Introduction
- Overview: Variables
- Create / Use Variables / Scope
- Overview: Data Types
- DT: Integer
- Overview: DT Fields/Methods
- DT: Floating-point
- DT: Decimal
- Floating-point Data Type
- Convert to Strings
- Decimal Data Type
- DT: Char / String
- DT: Boolean / Date/Object
- Char Data Type
- String Data Type
- Boolean Data Type
- Date Data Type
- Object Data Type
- Summary

**Variables / Data Types**
- Introduction
- Converting Data Types

*(Continued on page 2)*

# Learning to Program Using Visual Basic 2008



- Value vs Reference Types
- Converting Data Types
- Using Conversion Functions
- Using ConvertClass
- Using Parse
- Value / Reference Types
- Const. / Enum. / Struct.
- Constants / Enumerations
- Structures
- Operators
- Arithmetic / String Operators
- Comparison/Logical Operators
- Type Operators
- Operator Precedence
- Summary

**.NET Framework Classes**
- Introduction
- Base Class Library (BCL)
- Some BCL Namespaces
- Using Framework Classes
- Generating Random Numbers
- Retrieving Computer Info
- Generate Random Numbers
- Retrieve Computer Info Demo
- Working with XML
- Write XML
- Read XML
- File Input / Output
- Write to / Read from Files
- Managing Files
- Managing Directories
- Retrieve Drive Info
- StreamWriter/StreamReader
- FileInfo Class
- DirectoryInfo Class
- DriveInfo Class
- Summary

**Strings and Dates**
- Introduction
- Overview: Strings
- Comparing / Searching Strings
- Compare / Search Strings
- Modifying / Extracting Strings
- Modify Strings
- Extract Strings
- Formatting Strings
- Stringbuilder Class
- Format Strings / Dates
- Stringbuilder Class
- DateTime / TimeSpan
- DateTime Properties
- DateTime Methods
- TimeSpan Properties
- Summary

**The My Namespace**
- Introduction
- Overview
- My.Application
- My.Computer
- My.User
- Demo: MyNamespace
- Demo: MyApplication
- Demo: MyComputer
- Demo: MyUser
- Summary

**Branching**
- Introduction
- Branching in Code
- Conditional Branching
- If Statements
- Simple If statements
- TryParse Method
- Nested If Statement
- Test for Multiple Conditions
- Select Case Statement
- Test for Single Condition
- Select Case
- More Complex Select Case
- Repeating Code Blocks
- Unbounded Looping
- While Loop
- Do Loop
- Demo: While Loop
- Demo: Do Loop
- Summary

**Looping**
- Introduction
- For Loops
- Using For Loops
- Use Loops with .NET
- List Drives w/For Loop
- For Each Loop
- List Drives w/For Each Loop
- Compare For and For Each
- Unconditional Branching
- Using the Exit Statement
- Using Goto
- Exiting a For Loop
- The Goto Statement
- The Continue Statement
- Demo: Continue Statement
- Summary

**Introduction to Classes**
- Introduction
- Revisit .NET FW Classes
- View Help Documentation
- Classes as Templates
- Class Constructors
- Shared vs Instance
- Different Types of Members
- Which is Shared or Instance
- Creating Your Own Classes
- Objects as Things
- Creating Your First Class
- IntelliSense at Work
- Adding XML Comments
- Add an Overloaded Method
- Using the Class View Window
- Using the Class Designer
- Create w/ Class Designer
- Investigate the Created Class
- Add Code to Finish the Class
- Test the Class
- Object Test Bench
- Summary

**Working with Classes**
- Introduction
- Specific Issues with Classes
- Value vs Reference Types
- Initial Values
- Working with Null References
- Intro to Garbage Collector
- Releasing Memory
- Overriding Finalize
- Deterministic Finalization
- Demo: Deterministic Final.
- Override Object Class Func.
- ToString Method
- Manipulating Object Ref.
- Copy Ref. vs Copy Value
- Instance vs Shared Members
- Instance/Shared Members
- Modules and Classes
- Summary

**Properties**
- Introduction
- Overview
- Calculate Property Values
- Validate Property Values
- Pass Args. to Properties
- Demo: Read / Write Properties
- Demo: Calculated Property
- Demo: Validate a Property
- Demo: Passing Arguments
- Summary

**Methods**
- Introduction
- Overview
- Passing Arguments to Methods
- Passing Arguments
- Class Constructors
- Constructors
- Saving / Retrieving Information
- Demo: Saving Information
- Demo: Retrieving Information
- Summary

**Advanced Methods**
- Introduction
- Returning / Passing Arrays
- Returning an Array
- Pass a Parameter Array
- Optional / Named Parameters
- Using Optional Parameters
- Using Named Arguments
- Static vs Instance Members
- Using Static Variables
- Using a Static Field
- Using a Static Method
- Summary

**Inheritance**
- Introduction
- Overview
- Polymorphism
- Add Mem. to Derived Classes
- Explore the Sample
- Base Class Members
- Inherited Members
- Demo: Add Members to DC
- Overloading Derived Members
- Calling Base Class Members
- Demo: Overriding DM
- Demo: Overloading DM
- Demo: Calling BC Members
- Abstract Classes and Members
- Sealed Classes and Members
- Abstract Classes
- Sealed Classes
- Summary

**Interfaces**
- Introduction
- Overview
- Implementing an Interface
- Interfaces in the .NET FW
- Define an Interface
- Implement an Interface
- IComparable Interface
- Summary

**Organizing Classes**
- Introduction
- Overview

*(Continued on page 3)*

# Learning to Program Using Visual Basic 2008



- Partial Classes
- Nested Classes
- Namespaces
- Demo: Partial Classes
- Demo: Nested Classes
- Demo: Class Library Properties
- Demo: Add Ref. to Library
- Demo: External Assemblies
- Demo: Nested Namespaces
- Summary

**Introduction to Arrays**
- Introduction
- Working with Arrays
- One Variable vs Many
- Array Sample
- Visualizing Arrays
- Arrays Under the Covers
- Create and Fill Arrays
- Resize Arrays
- Assign References to Arrays
- Cloning an Array
- Arrays of Objects
- Multi-Dimensional Arrays
- Retrieving Data From an Array
- Pass an Array as a Parameter
- Arrays and Method Parameters
- Work with Method Parameters
- Auto-Array Parameter Rules
- Arrays in the .NET Framework
- String.Split Method
- Process.GetProcesses Method
- Summary

**Manipulating Arrays**
- Introduction
- Manipulating Arrays
- Sorting Arrays
- Sorting with IComparable
- More Flexible Sorting
- A Smarter IComparer
- Making it Easier
- Avoiding IComparer
- Summarizing Sorting Options
- Searching in Arrays
- Searching Arrays
- FindAll
- Summary

**Motivating Delegates**
- Introduction
- Motivating Delegates
- Demo: FileSearch0
- FileSearch1
- Demo: FileSearch1
- Event Interface Class
- IFileFoundHandler
- Adding Multiple Handlers
- Multiple Handlers
- Multiple Listeners
- Summary

**Introducing Delegates**
- Introduction
- Introducing Delegates
- Declare / Invoke Delegate
- Instance Delegate
- Shared Delegate
- Digging Deeper into Delegates
- Delegates with ILDASM
- Delegate v. MulticastDelegate
- Multicast Delegate
- Summary

**Events**
- Introduction
- Working with Events
- Events Like in VB6
- Demo: Raising Events
- Multiple Handlers
- Events with ILDASM
- Exceptions and Multiple Events
- Demo: Raising Events
- Manually Raising Events
- .NET Event Design Pattern
- Add/Remove Handlers Dynam.
- Summary

**Introducing Generics**
- Introduction
- Overview: Generics
- Generic Methods
- Generic Classes
- Demo: Swapping Values
- Demo: Generic Methods
- Demo: Generic Classes
- Summary

**Generics and Arrays**
- Introduction
- Overview
- Sorting Arrays
- Sort with IComparer
- Demo: Sorting Arrays
- Demo: Sort with IComparer
- Sorting with Generic Comp.
- Search w/ Generic Predicates
- Demo: Sorting w/GC
- Demo: Searching in Arrays
- Demo: Searching w/GP
- Summary

**Generic Interfaces/Constraints**
- Introduction
- Overview: Generic Interfaces
- Generic Constraints
- Generic Sorting Arrays
- Generic Sort w/ ICompare
- Generic Comparison
- Generic Compare Method
- Summary

**Generic Lists**
- Introduction
- Overview: Generic Lists
- Using an ArrayList
- Using a Generic List
- Sort with List Class
- Summary

**Handling Exceptions**
- Introduction
- Overview: Exception Handling
- Default Error Handling
- No Error Handling
- Error the User Sees
- Simple Try/Catch Block
- Unhandled Exceptions
- Using an Exception Object
- Error Bubbling
- Demo: Using Exception Object
- Demo: Display Cust. Msg.
- Catching Specific Exceptions
- Ordering Catch Blocks
- Trap Multiple Exceptions
- Code: Multiple Exceptions
- View Exception Handling Docs
- Summary

**Creating / Throwing Exceptions**
- Introduction
- Exception Handling Options
- Using the Throw Keyword
- Throwing Exceptions
- Demo: Throwing Exceptions
- InnerException Property
- Running Code Unconditionally
- Finally Block and/or Catch
- Finally Block
- Using Statement
- Creating an Exception Class
- User Defined Exception Handler
- Summary

**Generic List**
- Introduction
- Considering Data Structures
- Generic Collection Interfaces
- IEnumerable
- ICollection
- IList
- IDictionary
- Why Think About Interfaces?
- Interfaces Syntax
- Generic List Class
- List Behavior
- List Class Methods
- Demo: Generic List Class
- Return the List
- Demo: Return a List
- Summary

**List Sorting**
- Introduction
- Sorting the List
- Demo: Sort a List
- Look for Items in a List
- Demo: Look for Items in List
- Working with Predicates
- Predicates and Delegates
- List Methods using Predicates
- Test Predicates
- System.Action Predicates
- Converter Predicate
- Summary

**Other Collections**
- Introduction
- Dictionaries Stacks Queues
- Dictionary Class
- Demo: Dictionary Class
- SortedDictionary SortedList
- Demo: SortedDictionary
- Using Queues and Stacks
- Stack
- Queue
- Creating Collection Classes
- Create Custom Collection
- Summary

**Language Extensions**
- Introduction
- LINQ and Languages
- The Sample Application
- Demo: Without New Features
- Implict Type Delcarations
- Demo: With New Features
- Object Initializers
- Constructors to the Rescue
- Demo: Object Initializers
- Summary

**Lambda Expressions**
- Introduction
- Lambda Overview
- Using Delegates
- Demo: Delegates
- Predicate Delegate Type

*(Continued on page 4)*

# Learning to Program Using Visual Basic 2008

page 4

- Demo: Predicate Delegate
- Using Lambda Expressions
- Demo: Lambda Expressions
- Lambda and Delegates
- Lambda and Sorting
- Func Delegate Type
- Demo: Func Declaration
- Extension Methods
- Creating Extension Methods
- Demo: Extension Method
- Chaining Operations
- Demo: Chaining Operations
- Extension vs. Built-In Methods
- Anonymous Types
- Demo: Anonymous Types
- Summary