

Visual Basic .NET represents a radical change in the evolution of the Basic language. Some have called this newest version the “Javatization” of Visual Basic. There are certainly many similarities now between Java and Visual Basic .NET. Like Java, Visual Basic .NET is now compiled to a bytecode representation, with the resulting file run by the Common Language Runtime, which is very similar to the Java Virtual Machine. There is now a set of class libraries for Visual Basic .NET that provide much the same functionality as the base class libraries in Java. Microsoft® has also added full object-oriented programming functionality to Visual Basic, including full inheritance. Microsoft even included the ability to write Console-based applications in Visual Basic .NET, using either Visual Studio.NET or, and this came as quite a shock to me when I discovered it, using just a text editor to create the source file for compilation.

I have been teaching Visual Basic for the past four years and have been very disappointed in the introductory VB textbooks that have been on the market. Too many of them ignore teaching solid programming fundamentals in order to wow the student with the whiz-bang features of the user interface. After all the user interface features have been described, these books move right into showing the student how to design full-blown applications, again ignoring the fact that the reader has had at the most ten to eleven weeks experience with the language.

In the Fall of 2000, while lamenting the sad state of Visual Basic textbooks, I received the first beta version of Visual Basic .NET. I immediately recognized that this version of Visual Basic is a major shift in philosophy from Visual Basic 6 and its predecessors. At the same time, I decided it was time to write my own Visual Basic textbook using Visual Basic .NET as the target language. One of the first publishers I looked to was Jones and Bartlett. I have been using the Dale books on C++ and Java for several years and like how the Dale books focus on problem solving and program writing, testing, and debugging. When I contacted Michael Stranz, the Computer Science acquisition editor, he informed me that Jones and Bartlett was looking for someone to take the Dale series into Visual Basic. Voila. A relationship, and a book, this book, was born.

This book is *my* attempt to right the wrongs that *I* see in the Visual Basic textbook market. The focus of this book, as with all the Dale books, is on the fundamentals of programming. Each chapter focuses on one of the major programming constructs of the language. User interface details, while not ignored, are kept to just the basics of form design in Visual Basic .NET. I wrote this book feeling that once the reader has a good grasp of using Visual Basic .NET, the reader can venture out on their own to learn more about user interface design and development.

Chapter Coverage

Chapter 1 begins with basic definitions, computer concepts, problem-solving techniques, ethical issues, and a case study. We introduce objects from the very beginning, with their definition in this first chapter and the consistent use of object-oriented terminology.

In Chapter 2, we examine the String class (and data type) and aspects of its interface. Through the next several chapters, we introduce new library classes and point out their design features. We also tread carefully through the process of declaring a reference variable, instantiating an object, assigning it to a variable, and using the object. These are difficult but essential concepts for beginners to grasp. Our goal is for students to gradually develop a complete and deep understanding of what an object is, how it works, and what makes its interface well-designed. Then, by Chapters 7 and 8, students are ready to build significant and realistic classes.

Chapter 2 further covers sequential control flow, and simple output to a window using `MessageBox.Show`. The reader also learns how to use the Visual Studio .NET Integrated Development Environment (IDE) for entering and running Visual Basic .NET programs. In Chapter 3 we examine in more detail how to use Windows Forms and labels for output. Simple event handling is then introduced through the `Dispose` method for closing a form. The `Dispose` method is quite simple compared to other form methods and provides a good introduction to class methods.

In Chapter 4, we turn to the numeric types and expression syntax. We cover type conversions, precedence rules, the use of numeric types, and additional methods that can be applied to String objects. We also reinforce the distinction between the reference and primitive types in this chapter. Students gain further experience with form output and event handling in this chapter.

With Chapter 5, we take the next step in event handling by introducing the essential elements of an input dialog. Students learn how to use a textbox for data input and a button to signal when the textbox is ready. Visual Basic .NET makes this process easy since the event wiring is built-in, making it easier to write programs for events than it is in languages such as Java. The chapter closes with a discussion of object-oriented design that introduces the CRC card (Classes, Responsibilities, and Collaborators) as a mechanism for organizing an object-oriented design.

The primary goal of Chapter 6 is to introduce branching and the Boolean type. But the motivation for its use, in the end, is for students to be able to handle events from

multiple sources in a single event handler. Students have now learned all of the basic user interface elements necessary to write a wide range of interactive programs.

Chapters 7 and 8 are the heart of the text. In these chapters we bring together all of the informal discussion of classes and objects to formally introduce the mechanisms for defining new classes, methods, and derived classes. Students learn how to read the documentation for a class hierarchy and how to determine the inherited members of a class. They also see how the classes are related through Visual Basic .NET's scope rules. Namespace syntax is introduced so that multifile programs can be written and user-defined classes can be created and imported to other code. The key object-oriented concepts of encapsulation, inheritance, and polymorphism are treated thoroughly, and their use is demonstrated in the case studies.

User's who are familiar with procedural languages may wonder, at this point, "What happened to looping?" The answer is that event driven I/O has permitted us to postpone it to Chapter 9, where it naturally fits with file I/O and prepares students for the upcoming chapters on arrays. As always, we cover the basic concepts of looping using only the *While* loop. Object-oriented languages haven't changed the basic fact that beginners have to make conceptual leaps to understanding looping. Because there is more syntax to learn in an object-oriented language, it is tempting to relegate all those "old-fashioned" control structures to a single chapter and just breeze by them as if they are suddenly made easier in the presence of the additional complexity.

We find that students are still well-served by a careful study of the basic programming elements. We focus on how loops are used in algorithms while introducing only the minimum syntax necessary to illustrate the concepts. In that way, students don't develop the misperception that the different forms of control structures are bound to the different syntactic structures in a language. [This approach also avoids the situation that we commonly see in which a student is focused on their confusion over choosing among different looping statements when they are really still unsure of the underlying algorithmic mechanism that they wish to express.] For those instructors who feel strongly that they prefer to show students all of the control structure syntax at one time, Chapter 10 covers the additional branching and looping structures in a manner that enables appropriate sections to be covered as extensions to Chapters 6 and 9.

Chapter 10 is the "ice cream and cake" section of the book, covering additional control structures that make the coding of certain algorithms easier. In addition to the *Select Case*, *Do*, and *For* statements, Chapter 10 introduces the concept of exception handling. We show students how to use the *Try-Catch-Finally* statement to catch exceptions. Because we've already covered inheritance, it is a simple matter to define new exception classes that can be thrown between sections of user code. Students then are able to write code that is *robust in the face of errors* that cannot be handled directly with testing and branching.

Chapters 11, 12, and 13 are devoted to composite data structures. In Chapter 12, the basic concept of a composite structure is introduced and illustrated with the Visual Basic .NET array. In Chapter 13, we show how an array can be used to implement a general-purpose list class. Our prior class designs have been in the context of specific applications, and this is the first taste of an object-oriented design that does not have a predefined client. Then, in Chapter 14, we extend the discussion of arrays to multiple

dimensions, and through a case study we show how they can be used to represent mathematical matrices. Given this numerically motivated case study, it is also natural to review the limitations of floating-point numbers as they are represented in the computer.

One last programming topic, interfacing with a database, is introduced in Appendix A. This appendix is included to give the reader a taste of how Visual Basic .NET is used in the “real world”.

Chapter Features

Goals Each chapter begins with a list of learning objectives for the student. These goals are reinforced and tested in the end-of-chapter exercises.

Problem-Solving Case Studies A full development of a problem from its statement to a working Visual Basic .NET application is developed. In chapters beginning with 5, the CRC card design strategy is employed to develop object-oriented designs that are then translated into code. Test plans and sample test data are also presented for many of these case studies.

Quick Checks These questions test the student’s recall of major points associated with the chapter goals. Upon reading each question, the student immediately should know the answer, which he or she can verify by glancing at the answer at the end of each question so that the student can review the material in the event of an incorrect response.

Exam Preparation Exercises To help the student prepare for tests, these questions usually have objective answers and are designed to be answerable with a few minutes of work. Answers to selected questions are given in the back of the book, and the remaining questions are answered in the Instructor’s Guide.

Programming Warm-Up Exercises These questions provide the student experience in writing Visual Basic .NET code fragments. The student can practice the syntactic constructs in each chapter without the burden of writing a complete program.

Programming Problems These exercises require the student to design solutions and write complete Visual Basic .NET applications.

Case Study Follow-Up Exercises Much of modern programming practice involves reading and modifying existing code. These exercises provide the student with an opportunity to strengthen this critical skill by answering questions about the case study code, or making changes to it.

Supplements

Instructor's Toolkit

Also available to adopters on request from the publisher is a powerful teaching tool entitled Instructor's Tool Kit. This is a group of instructor teaching tools available for instructors; an electronic version of the Instructor's Guide, a computerized test bank, PowerPoint lecture presentations, and the complete programs from the text.

Programs

The programs contain the source code for all of the complete Visual Basic .NET applications and stand-alone classes that are found within the textbook. They are available as a free download for instructors and students from the publisher's web site: <http://computer-science.bpus.com/ubnet>. The programs from all of the case studies, plus several programs that appear in the chapter bodies are included. Fragments or snippets of code are not included nor are the solutions to the chapter-ending Programming Problems. These application files can be viewed or edited using any standard text editor. Most of the programs will run from the command-line compiler, but Visual Studio.NET is recommended for the beginning Visual Basic .NET programmer.

Student Lecture Companion: A Note-Taking Guide

Designed from the PowerPoint presentation developed for this text, the Student Lecture Companion is an invaluable tool for learning. The notebook is designed to encourage students to focus their energies on listening to the lecture as they fill in additional details. The skeletal outline concept helps students organize their notes and readily recognize the important concepts in each chapter.

A Laboratory Course in Visual Basic.NET

Written by Michael McMillan, this lab manual follows the organization of the text. The lab manual is designed to allow the instructor maximum flexibility and may be used in both open and closed laboratory settings. Each chapter contains three types of activities, Prelab, Inlab, and Postlab. Each lesson is broken into exercises that thoroughly demonstrate the concept covered in this chapter. The applications, application shells (partial applications), and data files accompanies the lab manual. They can be downloaded from the Jones & Bartlett web site.

Acknowledgements

The latest author to the Dale team is deeply indebted to the first three authors for providing him with an excellent text to work with. I'd say something about standing on the shoulder's of giants, but that phrase has certainly been overused lately. Nevertheless,

Nell, Chip, and Mark are all three excellent authors and instructors and have developed an ideal environment for learning the concepts of computer programming.

I can't thank the students at Pulaski Technical College who have taken my Visual Basic course over the past year enough. They were subjected to incomplete text, re-written text, and sometimes no text at all, but they stayed with me, taking me at my word that they were much better off learning Visual Basic .NET with an incomplete, but superior textbook. One student in particular, Karl Liss, read the text very carefully and pointed out several inconsistencies in the early drafts.

My fellow faculty members at Pulaski Tech, while not directly involved in the work, are always supportive and fun to work with. I want to especially thank my fellow "hall-mates", Raymond Williams, Bernica Tackett, and Beata Lovelace for all their support and good humor. I also need to say a special word of thanks to my division chairperson, David Durr, who not only listened to me complain quite loudly about the past state of VB textbooks, but also listened to me while I worked out some of the broader issues of teaching students Visual Basic .NET in an object-oriented manner.

The staff at Jones and Bartlett have been wonderful to work with. Michael Stranz has spent many hours with me, on the phone and asynchronously via email, discussing many issues concerning the making of this book. My production editor, Amy Rose, and her assistant Tara McCormick were patient with me as we finished up the book during the same time my school was right in the middle of finals, which is almost as trying a time for instructors as it is for students.

My thanks to the reviewers who helped to make this a stronger text: James Forkner, Pennsylvania State University; Thomas Gambill, University of Illinois; Donald Kussee, Utah Valley State College; James Prater, University of Alabama; Sandy Schleiffers, Colorado State University.

Last, but certainly not least, I need to thank my family—Terri, my wife; Meredith and Allison, my daughters; and Mason, my son, for being supportive of my work even though it meant long hours away from them while I typed away in my office. Hopefully, my children will learn from me how wonderful it is to be able to teach others how to do something you love to do so much.

M. M. M.