CS608 Lecture Notes

Visual Basic.NET Programming

Introduction to Visual Basic.NET

VB.NET Programming Environment (Review)

(Part I of IV)

(Lecture Notes 1A)

Prof. Abel Angel Rodriguez

CHAPTER 1 INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING 4

1.1 Understanding Object-Oriented Programming 4
1.1.1 The Procedural Programming Approach to Programming 4
Procedural Programming 4
Event Driven Programming 4
1.1.2 The Object-Oriented Programming (OOP) Approach 5
Thinking Objects 5
Data Encapsulation 5
Reusability 6

1.2 Components of an Object-Oriented Program 7
1.2.1 Understanding Classes & Objects 7
The Class 7
Objects 7
Private Data 7
Public Properties (Attributes) 7
Methods (Behavior) 8
Events 8
1.2.2 Creating Object-Oriented Programs (IMPORTANT!) 10
1.2.4 Object-Oriented Programming and Graphical Elements (Forms & Controls) 11

1.3 Object-Oriented Analysis, Design & Programming 12
1.3.1 Analysis and Design 12
1.3.2 Program Development Cycle 12
Visual Basic Solution & Project 12
Creating an Applications to solve a problem 13
1.3.3 Designing the Code – Creating an Algorithm 14
1.3.4 Summary of Strategy for Developing the Algorithm 17

CHAPTER 2 VISUAL STUDIO.NET DEVELOPMENT ENVIRONMENT 18

2.1 Microsoft .NET Framework and Visual Studio.NET 18
2.1.1 Microsoft .NET Framework 18

2.2 The Visual Studio.NET Environment & Visual Basics.NET 19
2.2.1 Introduction 19
Browser 20
Web Page or Web Application 20
2.2.2 Creating Project Using the Integrated Development Environment (IDE) 21
Startup Form/Startup Object 26

2.3 Visual Basics Modes, Error Types & Other Concepts 34
2.3.1 How Visual Basic Organizes Your Program or Application Files 34
2.3.2 Visual Basics Modes 34
2.3.3 Programming Errors 34
2.3.4 Two Aspects of Visual Basic Object Programming 35
2.3.5 Properties Revisited 35
Setting Properties at Design Time 35
Setting Properties at Run Time 36
Common Properties 36
2.3.6 Windows Applications Control Flow 37

2. 4. Visual Basic Debugging Tool 38
2.4.1 Understanding the Debugger 38
2.4.2 Setting Breakpoints 38

2. 4. Putting it All Together **39**
2.4.1 OOP Programming using Visual Basics In a Nutshell 39

2.5 Sample Programs **40**
2.5.1 Sample Program 1: Console Application – Login Request Application 40
2.5.2 Sample Program 2: Form-Driven Windows Application – Login Request Application 44
Three Step Process 46
2.5.3 Sample Program 3: Module-Driven Windows Application– Login Request Application Version 1 (Processing Code Inside Form) 50
Three Step Process 53
HOW IT WORKS: 58
2.5.4 Sample Program 4: Module-Driven Windows Application– Login Request Application Version 2 (Little or NO Processing Inside Form) (Best Practice!) 59
Three Step Process 61
HOW IT WORKS: 64
HOW IT WORKS: 65
2.5.5 Homework 65

Chapter 1 Introduction to Object-Oriented Programming

1.1 Understanding Object-Oriented Programming

1.1.1 The Procedural Programming Approach to Programming

- We will begin this course with a brief discussion of the programming methodologies that you are most likely accustom to in your previous Visual Basics.Net introductory courses.
- Programming as it was done in the past and still being done today in many cases is based on the ***Event-Driven*** and ***Procedural*** Programming approach.
- These methods of programming are based on what's known as **Structured Programming**. Structure programming has been the traditional way of programming.

Procedural Programming

- If you have taken a course in C, Visual Basic, Pascal, FORTRAN, Cobol etc. the programs you wrote were Procedural.
- In procedural programming, the **focus** of the programs was to **solve** a problem.
- For example, supposed you were asked to write a program to solve the following problem:

  - Write a Video Management System that will process the rental/return of videos tapes for a retail store such as a program used for Blockbuster Video.
- Using a language like C or can be done even with VB.NET, this is usually done as follows:

  1. Analyze the problem required to be solved: Design flow chart, algorithm etc.
  2. Break the problem into smaller manageable pieces, such as the Rental Module, Return Module, Customer Information Module etc.
  3. Design the UI for each of the pieces using Forms , controls or any structure supplied by the language to implement the UI
  4. Write code to implement each piece using variables, functions & procedures to implement each of the modular pieces.

  ❖ **Note that the focus is on solving the problem via the programming code and breaking the problem into smaller manageable pieces**.

- Dividing a program into Procedures/functions and modules is one of the cornerstones of structured or procedural programming. But here as some of the drawbacks:
  - As programs grow larger and more complex, even the procedural programming approach begins to show signs of strain. Projects can become too complex, schedules slip, more programmers are added, cost skyrockets etc.
  - In **Procedural programming** data or the variables & data structures that hold or store the data, are usually **unprotected** and may be accessible to functions & procedures that have no business changing them, therefore they can be easily corrupted.
  - Procedural programs are difficult to design because their chief components, procedures, functions and data structures don't model the real world very well.

Event Driven Programming

- If you wrote the Video Management Program using Visual Basics 6 or in some cases VB.NET, as it's taught in courses such as CS101 & CS508, then you would normally tend to write this program as an ***Event-Driven*** Application.
- ***Event-Driven*** applications react to user events or actions such as clicking buttons, check boxes or navigating through forms or graphical front-ends. These programs are still based on the procedural programming philosophy, but are base on code reacting to user actions on the GUI or front-end.
- The steps to write an Event-Driven program are as follows:
  1. Analyze the problem required to be solved and derive the algorithm:
     - Design flow chart, algorithm to solve this problem etc.
  2. Use Forms & Controls to designed the User Interface (UI)
     - Drop some controls to implement the GUI, such as labels, text boxes. Command buttons etc.
     - Use the controls to implement features such as Rental, Return, Customer Information, Video Tape Information etc.

3. Placed programming code inside the ***Event-Handlers of the*** controls on the Form, to respond to actions taken by the users on the controls. Such as the *button_Click()* event of a Command Button etc.

❖ **Note that with this approach, the focus again is on breaking the program into sections and solving the problem via the Form, controls & code in the Event-Handlers**

1.1.2 The Object-Oriented Programming (OOP) Approach

Thinking Objects

- The newer programming languages use a different approach. With OOP, programs are based on **real world objects**.
- This method of programming is based on creating programming code that emulates real world entities, thus the word Object.
- In OOP, instead of focusing on solving the problem, **you focus and think in terms of the Objects that will play an important role in the program**.
- That is you first create the ***Objects*** that are the important characters in the program, such as Employees, Departments, Customers, Products etc.
- Using ***Objects***, allow programs to be based on real world entities, such as a car, person, employee, customer, inventory part etc
- Examples of pure OOP languages are **C++, Visual Basics.NET & Java**.

- In OOP, each object has its own ***Properties*** or ***Attributes*** (access to Data), the ***Methods*** (Functions/Procedures) that operate on the data & the ***Events*** that are automatically triggered when the object is used or manipulated.
- The fundamental idea behind object-oriented languages is to combine into a single package both the ***data***, ***Methods*** (functions/procedures) & ***Events*** (Event-Procedures) that operate on that data. **Such unit is called an *object***.
- Combining the ***Data***, ***Methods*** & ***Events*** that operate on that data into a single package means that the ***Objects*** handle themselves and have a life of their own, and most important, they can be ***re-used*** in other applications
- The mechanism to implementing Object-Oriented Programming is the **Class** & the **Object**.

- Object-Oriented approach to solving the *Video Management* problem:

1. Analyze the problem required to be solved and derive the algorithm:
- Design the Objects that are the key protagonists of the program.
- For example, a Video Object, Customer Object, Employee Object etc.

2. Implement or create the **template or Classes** for each of the required Objects with the properties, methods (actions) and events required to perform the functionality of each object. For example implement a video object that behaves as a video, a customer object that behaves as a customer & an employee object that behaves as an employee.

3. Use Forms & Controls to designed the User Interface (UI) for implementing the Video Rental/Return processing

4. Create the **Objects** and use programming code to manipulate the Objects as necessary via the User Interface Forms to **solve** the problem at hand.

❖ **Note that with this approach, the focus is on the Objects not the problem. The object was the first thing that was created, then the problem was applied to the objects**

Data Encapsulation

- A very important feature of OOP is Data Encapsulation or Data Hiding.
- In OOP, the object's data is ***Private*** thus hidden and is only accessible by the ***Public*** *Methods* (Functions/Procedures) and ***Public*** *Properties*.
- ***Private*** data means that there is no way for the outside world to access the data directly. Thus the data is protected and invisible or hidden from the outside world.
- ***Public*** Methods & Properties are the interface or vehicle for the outside world to be able to access or manipulate the data.
- This means that you can only do to an object what the ***Public*** *Methods* and *Properties* allow you to do. If there is no Public Methods or Properties for a particular task, then it can not be done.
- An Object behaves exactly as they were specified by the Public Class Methods and Properties. No more, no less
- A benefit of Data Encapsulation is Robustness or a solid, reliable error-free Object.