An Introduction to .NET for MPE People

Rich Trapp
Senior Consultant
Managed Business Solutions

Introduction

- In the HPe3000 community, organizations have been running business critical applications on the HPe3000 over the last 20 years
- HP announced the sunset of the HPe3000 in 2001
- The organizations in the HPe3000 community have been developing a transition plan for migrating their applications off the HPe3000

Introduction

- The options for transition are:
  - **STAY:** Remain on the existing HPe3000 platform with the existing application(s)
  - **PORT:** Move the existing application(s) as is to a new platform
  - **BUILD:** Re-write or re-engineer the application(s) on a new platform, often enhancing the applications significantly
  - **BUY:** Purchase an off-the-shelf application package to replace the functionality of the existing applications

Introduction

- .NET may be involved if the organization chooses:
  - **PORT:** Some Port tools convert the existing applications into .NET; further enhancements may involve .NET development
  - **BUILD:** The application(s) may be re-written or re-engineered in a .NET development environment
  - **BUY:** Many off-the-shelf packages are now implemented in a .NET development environment; customizations and interfaces may best be written in .NET
- .NET is likely to be in your organization's future

Introduction

- In the HPe3000 community, the current IT staff is skilled in:
  - COBOL
  - 4gl's, such as COGNOS or Speedware
  - Image
  - MPE
- This staff will likely need to transition their skill set to .NET
  - VB.NET
  - C#
  - ADO.NET
  - SQL Server or other relational databases
  - .BAT files and the Windows scheduler

Agenda

- This presentation discusses
  - The major differences between the HP3000 and .NET
  - The benefits of .NET
  - Getting started with .NET

Major Differences Between the HP3000 and .NET

- The major differences between the HP3000 and .NET development environment are
  - The IDE
  - Object Oriented Design (OOD)
  - Relational Databases
  - Jobs and Job Scheduling

Integrated Development Environments (IDE)

- An Integrated Development Environment (IDE) is an application that allows for comprehensive development of application source code
- For .NET, this is Visual Studio.NET
- The IDE replaces the role of
  - QEDIT
  - The COBOL compiler
  - The 4gl compiler / interpreters
  - Screen designers, such as VPlus
  - Debuggers

Integrated Development Environments (IDE)

- Features of Visual Studio.NET
  - **Visual Screen Layout:** draw the screen instead of typing characters on a 24 x 80 screen
  - **Code Generation:** the drawn screens automatically create code behind the scenes to support itself
  - **Objects Library:**
    - File management
    - Date / Time
    - Array objects
    - Hash tables
    - Lots and lots of other routines
  - **Source Level Debugging:**
    - Call stack: keeps track of all of the method calls
    - Step-by-step view of source code execution

Integrated Development Environments (IDE)

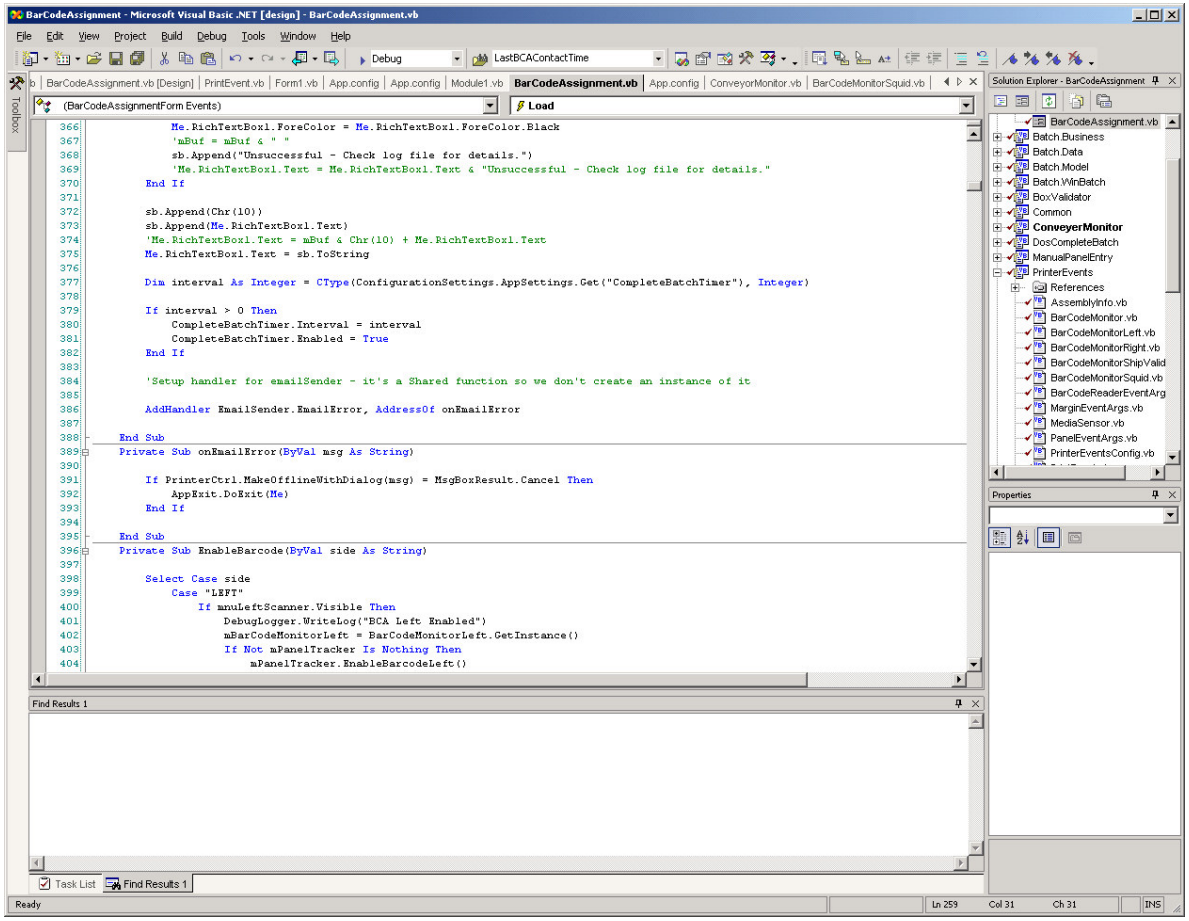
- Features of Visual Studio.NET
  - **Auto Formatting:** automatically formats your source code
  - **IntelliSense:** if working with a reserved word or object, Auto Complete shows the valid options based on what's typed so far
  - **Integrated Help:** highlight a reserved word and get a helpful description of it
  - **Go To Definition:** right click on a word or procedure code and the IDE takes you to that source code
  - **Highlights:** color coding of
    - Reserved words
    - Objects
    - Methods

Integrated Development Environments (IDE)

- Visual Studio.NET Module Structure
  - Visual Studio.NET collects classes into projects
  - Visual Studio.NET collects projects into solutions
  - These items comprise:
    - Executables (.EXE)
    - Libraries (DLL's)
  - As an example, a typical project might have:
    - 5 DLL's
    - 5 EXE's
    - Established as 1 solution and 10 projects

Integrated Development Environments (IDE)

- Sample Screen of Visual Studio.NET



Object Oriented Design (OOD)

- The primary difference between development in an HP3000 environment and a .NET environment is:
  - The HP3000 is geared toward procedural design
  - .NET is geared toward object oriented design

- Procedural Design:
  - Procedures or steps are established as a sequence of commands, acting on data structures

- Object Oriented Design:
  - Developers model real-world situations and business scenarios as objects that perform actions, have properties, and trigger events .

Object Oriented Design (OOD)

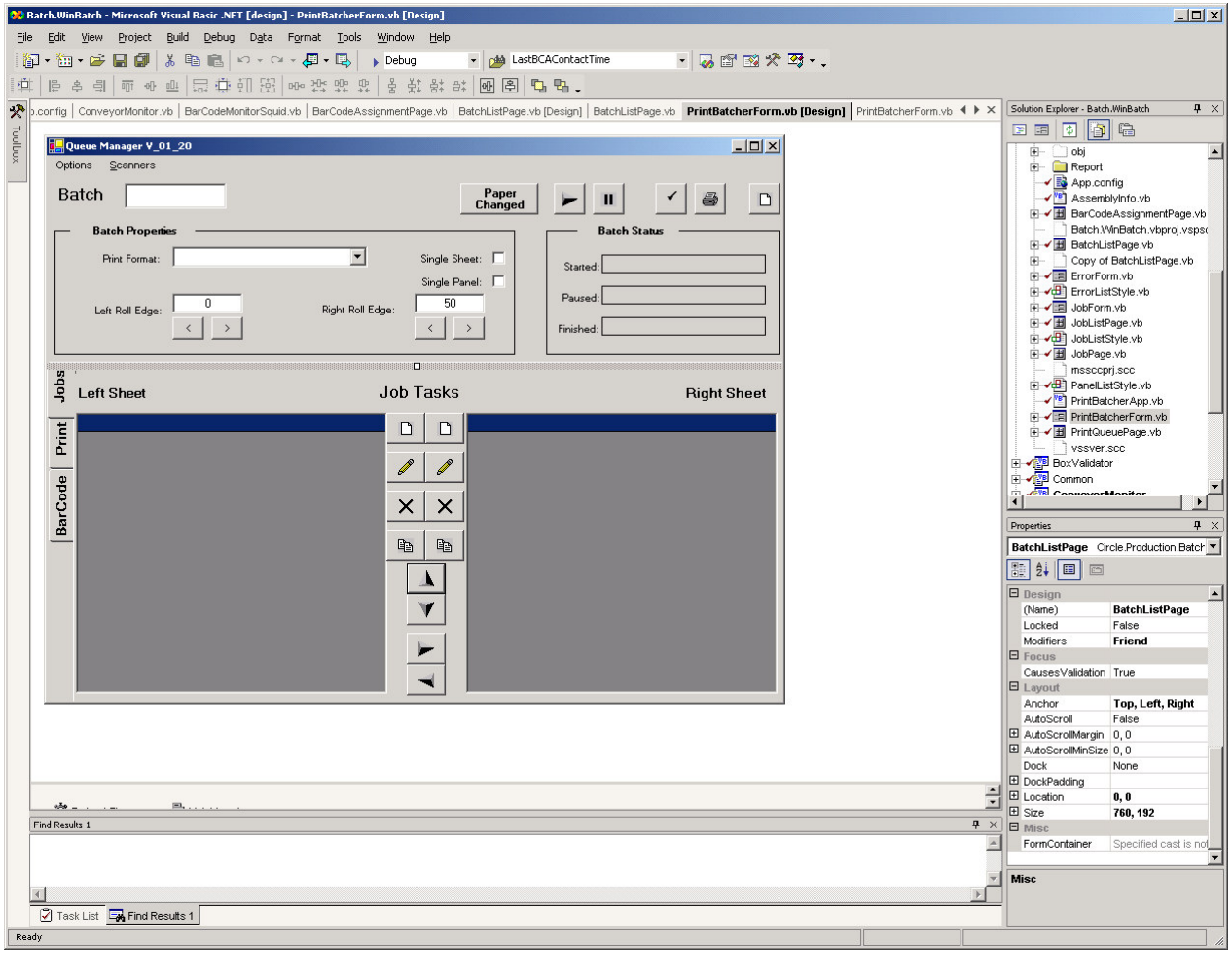
- Object Oriented Design contains the following concepts:
  - Classes: Definitions of common objects (i.e. a data type), to include the structure(s) of the data, and the procedures that act on that data
  - Methods: The procedures that act on the data
  - Objects: An instantiation of a class (i.e. a variable)
  - Properties: A member of a class that can implement “get” and “set” accessors and can be used like a variable
  - Shared Classes: Classes that do not require instantiation

Object Oriented Design (OOD)

- Classes provide inheritance
  - Sub-classes belong to classes
  - A sub-class inherits all data structures from its parent class
  - A sub-class inherits methods from its parent class
- Inheritance is important because
  - It promotes software re-usability
  - It standardizes coding approaches across the entire application environment
  - It reduces software development time

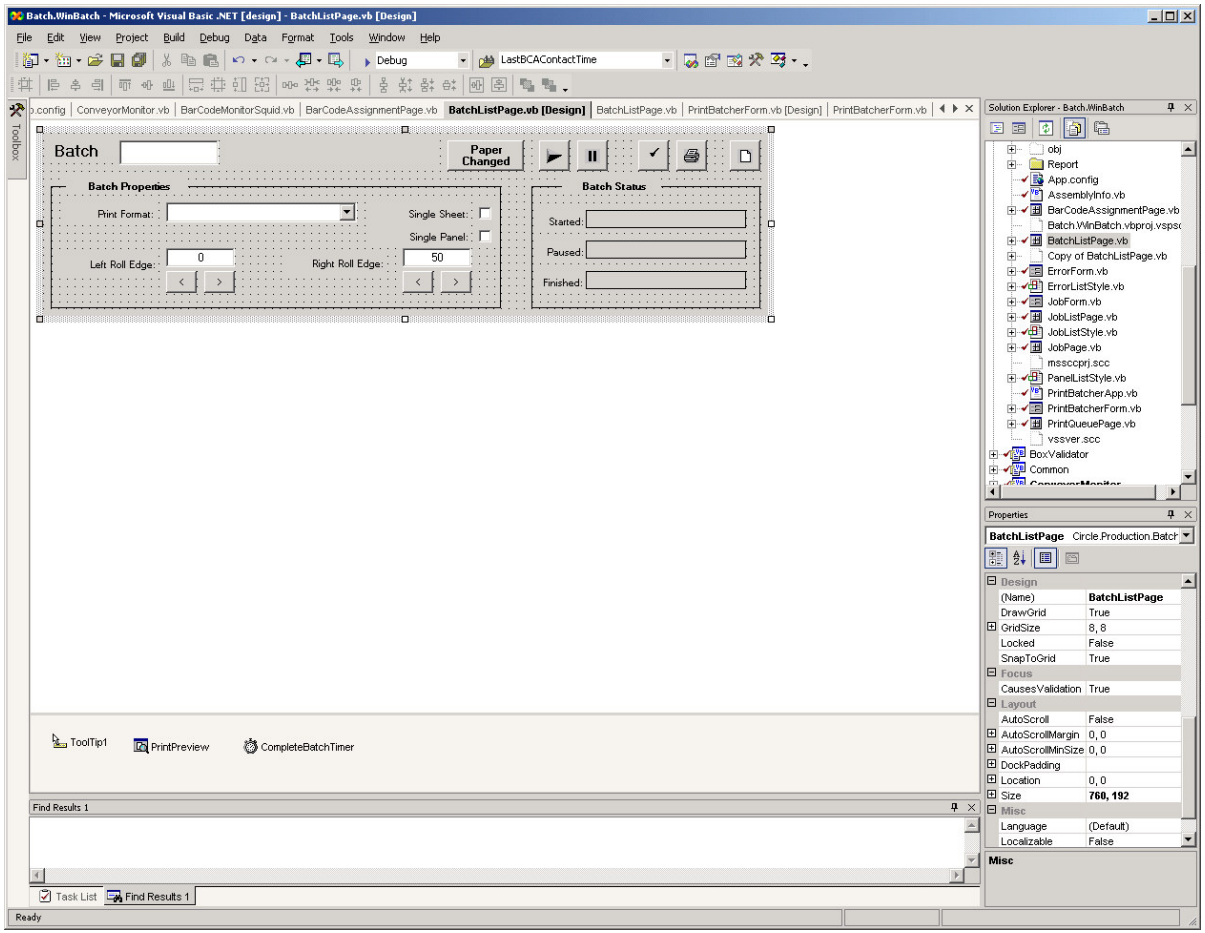
Object Oriented Design (OOD)

- Example: Class inheritance—a form



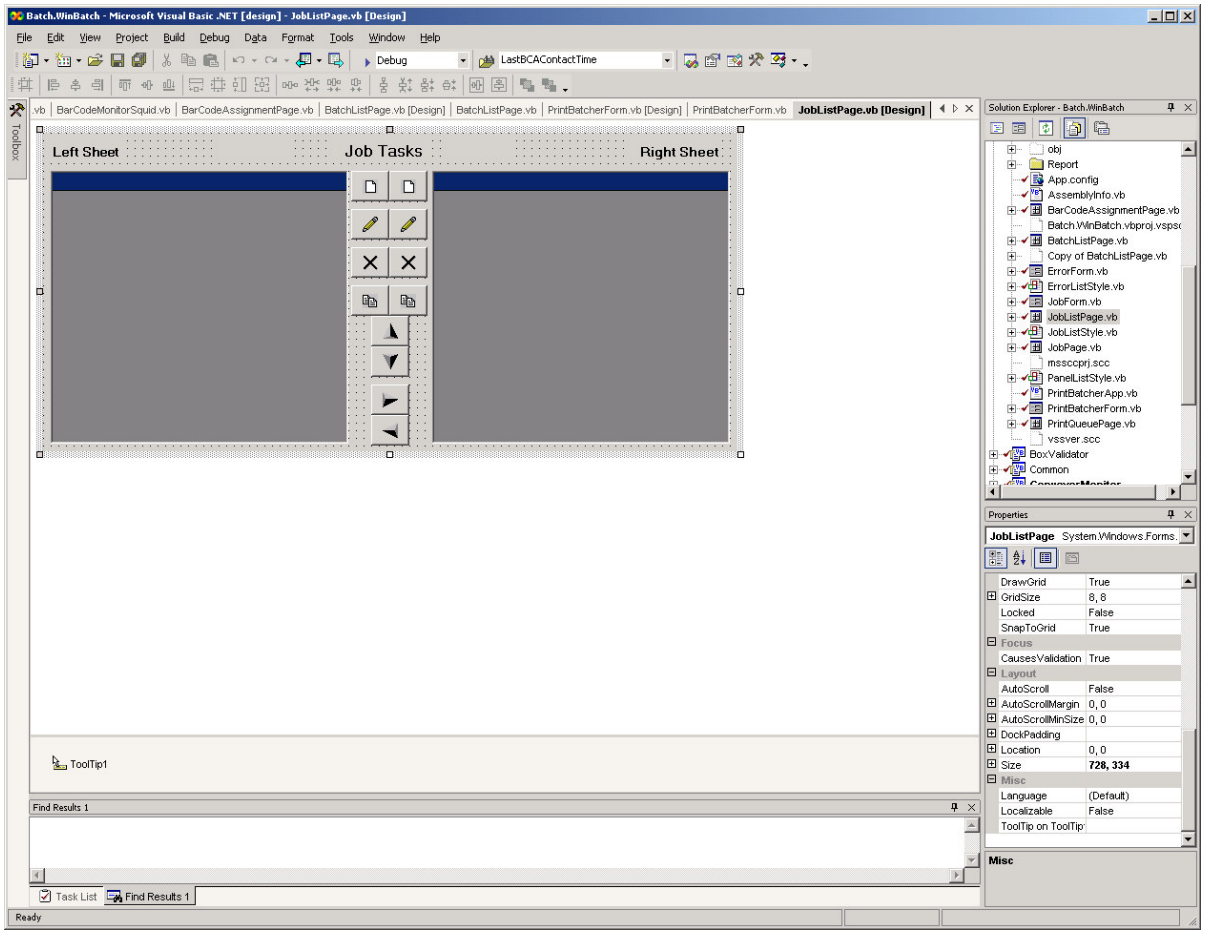
Object Oriented Design (OOD)

- Example: Class inheritance—a form



Object Oriented Design (OOD)

- Example: Class inheritance—a form

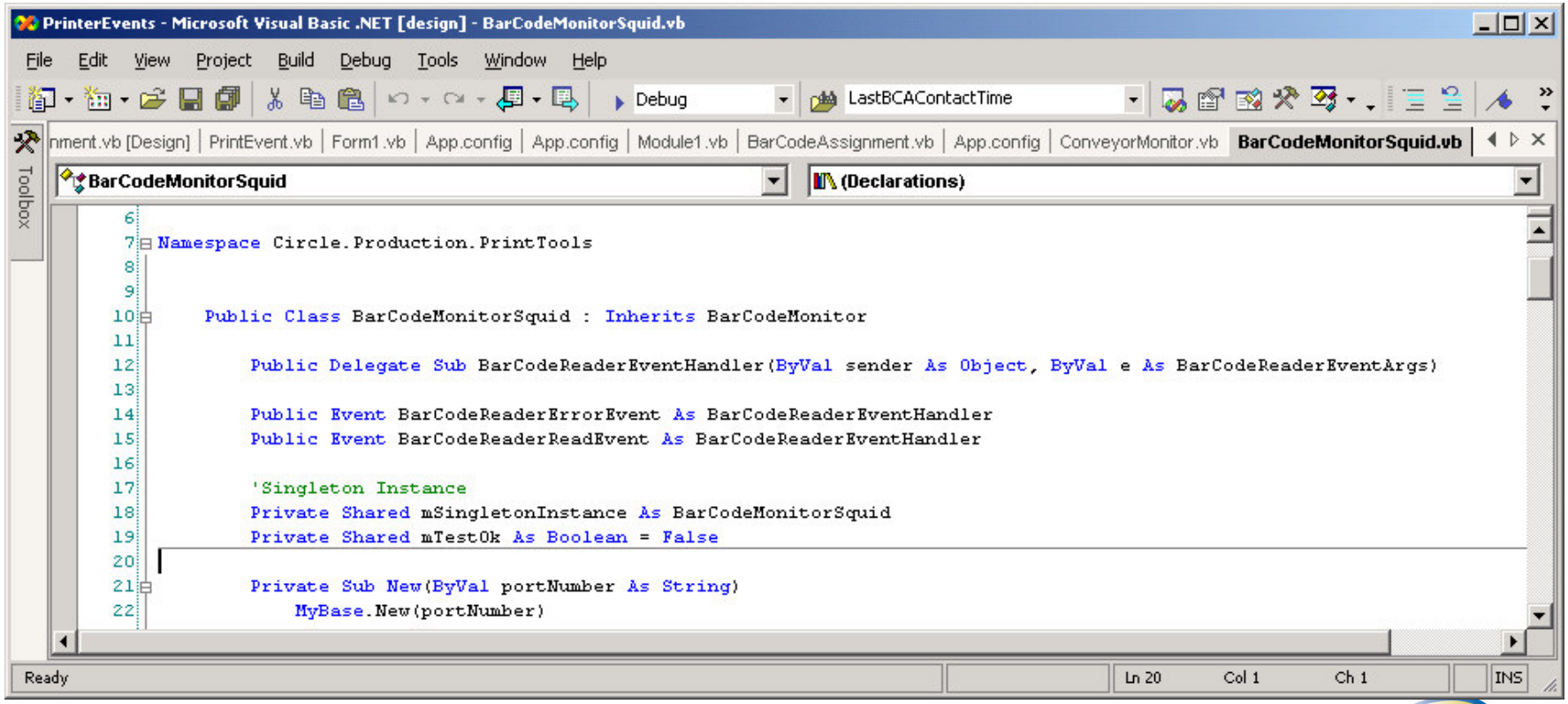


Object Oriented Design (OOD)

- OOD includes the use of **Events**
  - Events are raised by a method
  - Events are handled by calling objects
  - Events allow objects to focus on their own task and to notify calling methods of issues
  - Events prevent called objects from having to handle issues that are beyond their scope
  - This eliminates direct calls from one procedure to another, when they are unrelated
  - This eliminates ‘spaghetti code’

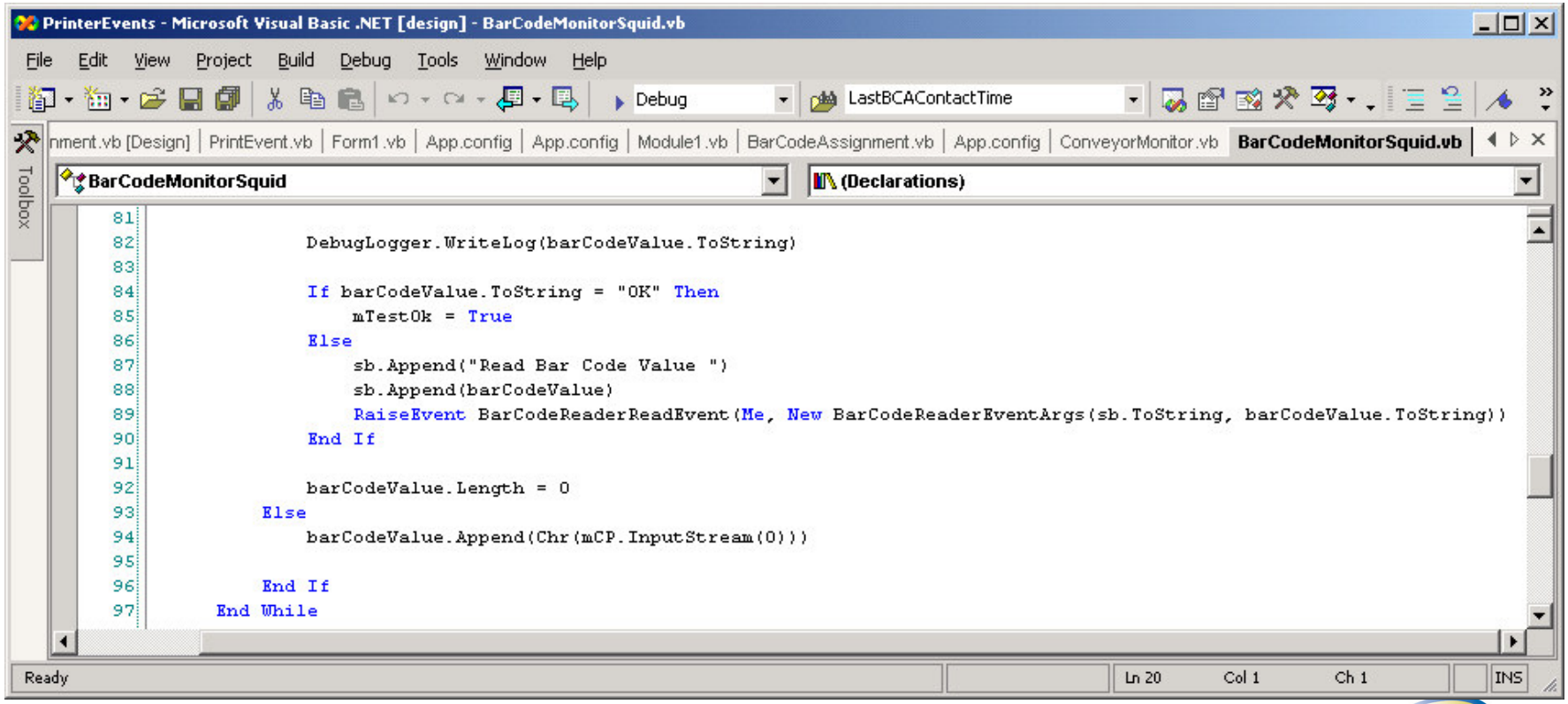
Object Oriented Design (OOD)

- Example: Event raising and handling



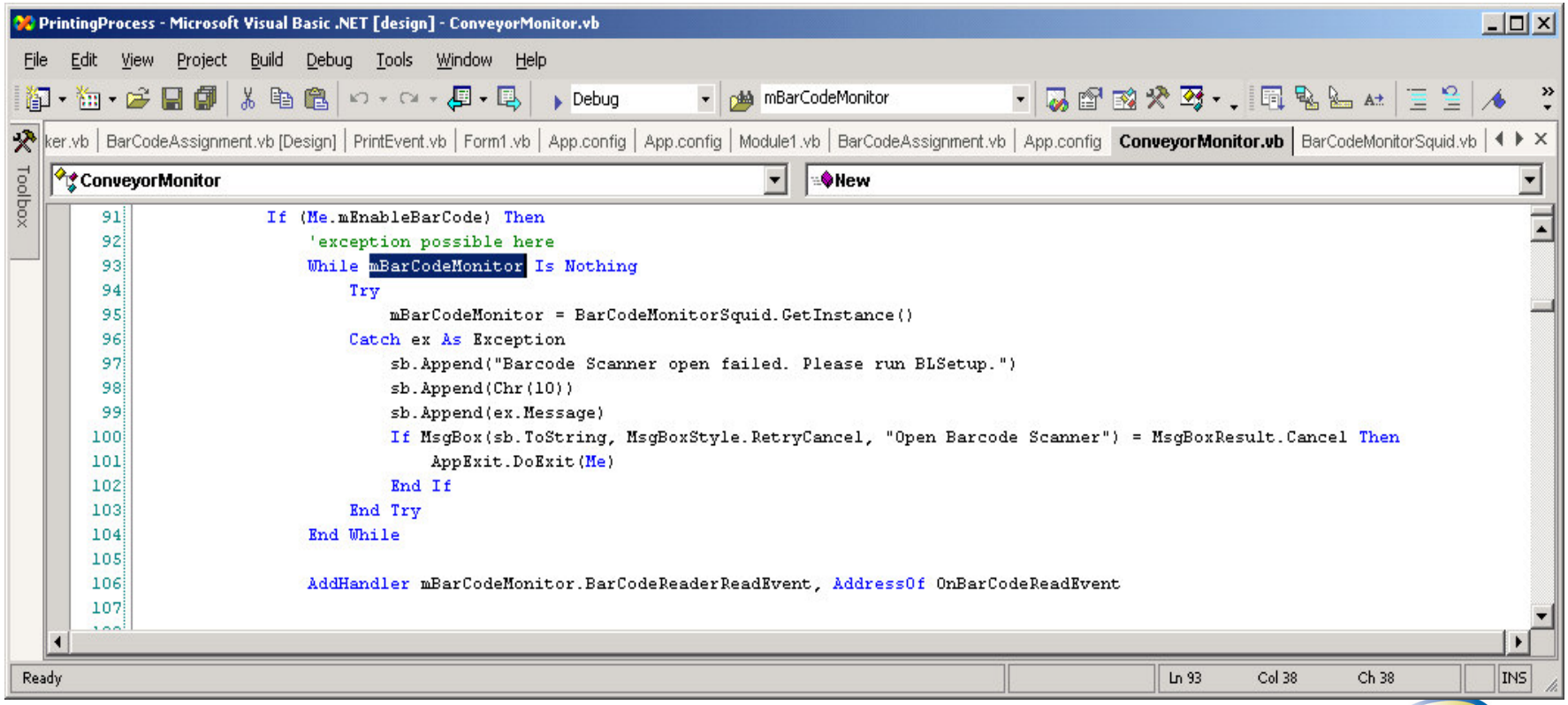
Object Oriented Design (OOD)

- Example: Event raising and handling



Object Oriented Design (OOD)

- Example: Event raising and handling

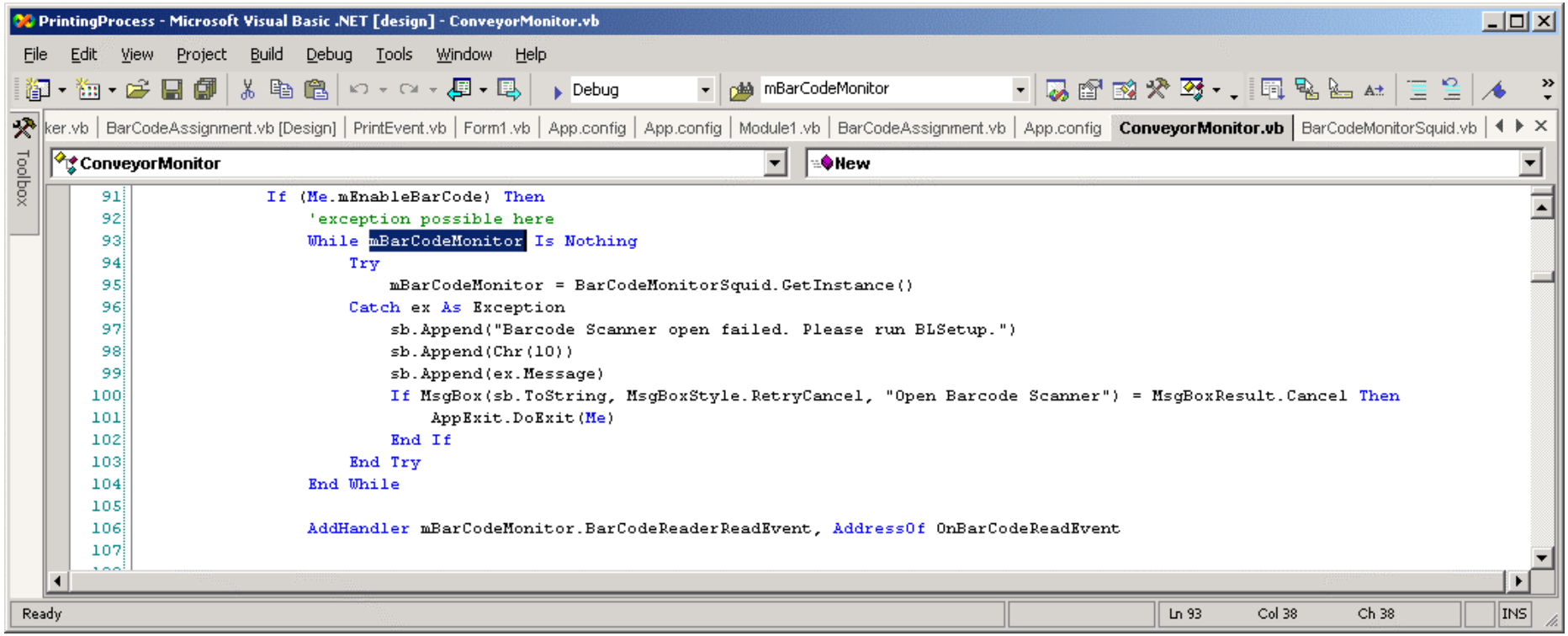


Object Oriented Design (OOD)

- OOD includes the use of **Exceptions**
  - Exceptions are raised by a method
  - Exceptions are handled by calling objects
  - Exceptions allow objects to focus on their own task and to notify calling methods of issues
  - Exceptions prevent called objects from having to handle issues that are beyond their scope
  - This eliminates direct calls from one procedure to another, when they are unrelated
  - This eliminates ‘spaghetti code’
  - Unlike events, exceptions use a Try, Catch, Clean Up code structure

Object Oriented Design (OOD)

- Example: Exception raising and handling



Object Oriented Design (OOD)

- There are no pointers in .NET!
  - (OK, there are, but you can't see them)
  - Objects are instantiated with a name
  - .NET is responsible for its own garbage collection

Relational Databases

- Image is not readily available in a .NET environment
- The databases used with .NET are typically relational:
  - SQL Server
  - Oracle

Relational Databases

- Image is a Network Database
  - Allows Master to Detail relationships
- Relational Databases allow
  - A relationship to be established between any two (or more) tables
    - Provides greater data modeling flexibility
    - Encourages normalization of the data
    - Improves the maintainability of the applications

Relational Databases

- .NET provides classes and methods that
  - Load database records into data structures
  - Associate screen fields directly to database records without having to write SQL
  - Handle transactions and rollbacks
- Database Administrators are required for relational database packages

Jobs and Job Scheduling

- Many kinds of job schedulers are available
  - Windows
  - SQL Server
  - Off-the-shelf
- These schedulers can invoke .NET programs directly, eliminating the need for ‘jobs’ in many cases
- Script files can be created for required system functions
- The biggest issue is tracking which schedulers are running which jobs

Major Differences Between the HP3000 and .NET

- .NET is a very different environment from the HP3000
- The biggest difference is the object oriented environment instead of the procedural environment
- .NET is supported by very good tools
  - The IDE
  - Classes, methods, and objects
  - Event and exception handling
  - Databases
  - Job schedulers

Getting Started with .NET

- Learning .NET requires:
  - Training
  - Experience

- Experience is crucial to the learning process

Getting Started with .NET

- Training: Good Books
  - Microsoft's "Visual Basic .NET Step by Step"
  - Microsoft's "Visual Basic .NET Core Reference"
- Books by MBS'er Kevin Hoffman:
    - "Professional .NET Framework, by Wrox Press
    - "Professional ADO.NET", by Wrox Press
    - "C# Programming Evolution", by SAMS Press
    - "Visual C#.NET 2003 Unleashed", by SAMS Press (forthcoming)
- Training: Pitfalls
  - The books are often geared to a VB6 audience, not an HP3000 audience
  - The books often contain simplistic examples that underplay real world complexity

Getting Started with .NET

- Training: Understanding The CLR
  - CLR stands for Common Language Runtime
  - The CLR functions in the background
  - The CLR does its job; programmers do not generally have to worry about it
  - All .NET books open with a discussion of the CLR

Getting Started with .NET

- To gain experience with .NET, follow these steps
  - Use the IDE
  - Use the debugger
  - Learn the libraries
  - Understand OOD
  - Get experience with .NET ‘quirks’

Getting Started with .NET

- Experience: Use the IDE
  - Learning the editor and debugger is easy
  - The biggest issue is that the number of modules (classes) in an application is intimidating
- Experience: Use the debugger
  - Start with a sizable application, and walk through the execution of the source code using the Visual Studio.NET Debugger
  - The syntax is not too hard to understand
  - Biggest issue is understanding ‘How did I get here?’

Getting Started with .NET

- Experience: Learn the libraries
  - There is a tremendous number of objects provided by Microsoft
  - Help and F1 are your friends
  - Internet sites, such as Google, can help programmers find ones that are needed

Getting Started with .NET

- Experience: Understand OOD
  - Becoming used to class inheritance
  - Keeping track of the call stack
  - Need to view classes and subclasses as ‘superimposed’ code
    - Sometimes a parent class will call a subclass’s method
    - Sometimes a subclass will call a parent class method
  - Understanding that everything is an object
    - For example, strings now have methods associated with them
  - Understanding that events control the programs (e.g. user events), not the procedures
  - Hardest part of learning .NET

Getting Started with .NET

- Experience: Get experience with .NET ‘quirks’
  - War Story: Literal over-written by garbage collection

Getting Started with .NET

- The greatest similarities between HPe3000 and .NET will be:
  - Project deadlines
  - Programming and logic skills
  - Problem solving skills
  - Figuring out why something doesn't work
- But otherwise, it's a completely different environment
- Plan on 3 – 6 months to become functional in .NET

Benefits of .NET

- The IDE provides efficient development, code generation and extensive on-line help
- OOD enforces excellent programming standards
- OOD allows for localized maintenance and enhancements in the future
  - Eliminates interdependence of procedures

Benefits of .NET

- There is an extraordinary amount of tools and libraries available
  - Microsoft provided objects and libraries
  - Free objects and libraries on the web
  - Interfacing mechanisms, such as XML
- Application integration is becoming easier
  - Web services
  - Standardized interface formats
  - There is no longer a need for fixed format files

Conclusion

- .NET will be a major player in the development of applications well into the future
- There is a growing market share, resource base, and material available for .NET development
- This is occurring because .NET provides
  - Efficient development
  - Well structured applications
  - A large number of interfacing techniques and interfaces
  - A large quantity of existing, re-usable source code

"Wherever you go, there you are."

- Buckaroo Bonzai

HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP