Structured Exception Handling

YONG ZHANG

Structured Exception Handling

- Application errors happen
- Avoid crashes
- C# deals with error handling through **structured exception handling**
  - Wrap code in try/catch/finally blocks
  - Monitor unexpected events
- Example: 03-ExceptionHandling/StructuredExceptionHandling
- Ctrl + . To open the quick fix
- Debug.WriteLine

Exception Flow

- Flow:
  - The framework checks to see if the function handles the exception
  - If not, pass the info up the call stack, until it finds an exception handling routine.

- Example: 03-ExceptionHandling/ExceptionFlow

Throwing Exceptions

- Using the throw keyword is a great way to rethrow an exception, backup the call stack so that it can be handled someplace else in the application.
- Primarily, you might use that if you have a specific set of classes that are designed to handle exceptions and write that information out to a log file.
- Example: 03-ExceptionHandling/ThrowingException

Exception Classes

- Open Object Browser, find System.Exception, explain
  - The class
  - The Data member variable
  - HelpLink
  - innerException
  - Message
  - Source
  - Three Exception() constructors
  - Search for DivideByZero exception
  - Search for IOException

- Example: 03-ExceptionHandling/ExceptionHandlingExample

Exercises – Grade Project

- Modify Name property in GradeBook class (line 64)
- Modify main program to set Name with an empty string to show the exception.
- Handle the exceptions in Program class using try/catch
- Add a grades.txt by adding a new item to the project and change the property of this file to copy if newer to output directory, add some grades to the file
- Add code to read the text file,
- Change the file name to introduce exception
- Add try/catch block
- Try to use FileStream to demo hwo to clean up the resource in Finally block: Add Finally block to clean up the resources
- Use using statement with resource handlers