Programming with C#

# Exception Handling Practice

1. Open Visual Studio if it is not already open.
2. Locate the folder where you extracted the companion content.
3. In the Chapter 6 folder, locate either the Windows 7 or Windows 8 folder.
4. Open the MathsOperators project.
5. On the Debug menu, click Start Without Debugging or press CTRL F5.
6. Visual Studio will build and run the application. You will input some invalid values into the inputs and see the errors that occur.
7. Type your name in the Left Operand box and an integer value in the Right Operand box.
8. Click the Addition button and then Calculate.
9. The application will display a dialog indicating that something went wrong. This is an example of an unhandled exception in the application.
10. Stop the application and return to Visual Studio.
11. This time, press F5 or select Start Debugging.
12. Enter the same information as before and press Calculate.
13. This time, Visual Studio traps the exception, because you started in debug mode.
14. Visual Studio will highlight the line of code that generated the exception. The exception was generated because you entered a text value when an integer value was expected. The Parse method throws an exception if it cannot cast the value to the requested data type.
15. Experiment with the Actions available on this dialog box, such as View Detail, to see more information on the exception that was thrown.
16. Stop the application and return to Visual Studio.
17. Open the code for MainWindow.xaml.cs.
18. Surround the code that parses the input with a try block. Ensure that you enclose all the code from the ***int lhs*** line to the ***result.Text*** line.
19. Create a catch block to catch a FormatException and output the exception Message in the result text box.
20. Start debugging or press F5.
21. Enter the same information you did previously that resulted in the exception and press Calculate.
22. Your exception handling code should execute and output the exception message into the results text box.
23. Stop the application.
24. You will now change the exception handling code to work with exception propagation.
25. Go back to the MainWindow.xaml.cs file and remove the try and catch code that you entered previously.
26. You will still generate an exception with this code but rather than handling it here, the exception will be propagated back up the stack to the calling method.
27. Locate the calculateClick() method.
28. Surround the if and else if statements with a try block.
29. Create a catch block that will catch a FormatException and output the exception message to the result textbox.
30. Start debugging and enter the same information as before and click Calculate.
31. The exception should be generated and handled, with the results being displayed in the result textbox again.
32. Test each of the mathematical methods to ensure the exception is caught by each one.
33. When the application performs correctly, stop the application.
34. In the next set of steps, you will write code that will throw an exception based on a potential user error.
35. In the existing application, there is no code that verifies an operation option is selected. As a result, you can run the application, enter values, click Calculate, but nothing will display in the result box and no exceptions are generated.
36. To handle this, you will add some code to check this condition and throw an exception.
37. At the end of the current if-else statements, add an else block that will be the fall through if none of the operation options are selected. In this else block, throw a new InvalidOperationException passing in text indicating that no operator was selected.
38. Test the application by pressing F5 and note that Visual Studio will detect the exception.
39. Stop debugging.
40. Return to the code file and add another catch block to the existing catch blocs that will catch this InvalidOperationException.
41. In the catch block, set the value in the result textbox to the exception message.
42. Test the application and ensure that your exception is caught and the results are displayed in the text box.
43. Stop the application and return to the code file.
44. To complete the catch block section, add a catch block that will catch any other non-specific exception and output any message to the result text box.
45. Select the File menu in Visual Studio.
46. Select Close Solution.