Name intentionally omitted due to FERPA

Professor Chukwuemeka

Visual Basic – Final Exam

April 24, 2021

Microsoft 98-361 Questions

I will be completing the following questions from Microsoft 98-361. 18, 33, 43, 107, and 113. The website they can be found on is: **https://www.itexams.com/exam/98-361**

Question 18:

The purpose of the Catch section in an exception handler is to:

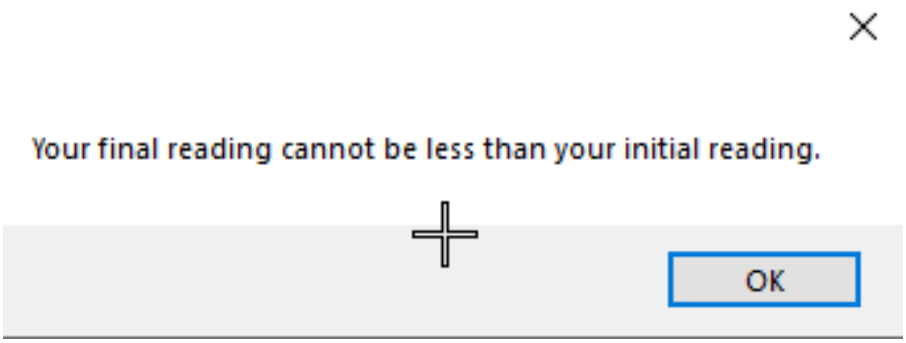
A. Break out of the error handler.
B. Conclude the execution of the application.
C. Execute code only when an exception is thrown.
D. Execute code regardless of whether an exception is thrown.

Answer : **C**

My Explanation: Using the catch system is great for error handling, as it essentially looks for and grabs any error that is thrown by a problem – whether it be improper data entry by the user or an error in the code. The catch system is commonly used for error-handling or in order to show the program developer what the error is that is causing their program to crash.

Ex. Of error handling

```
    'The catch statement looks for any exceptions thrown by the program
Catch ex As Exception
    'This forces the caught exception to be shown in a message box with an "OK" prompt
    MessageBox.Show(ex.Message)
    'Once the user clicks "OK" on the prompt, the program closes
    Close()
```



Question 33:

How does a console-based application differ from a Windows Forms application?

A. Console-based applications require the XNA Framework to run.
B. Windows Forms applications do not provide a method for user input.
C. Windows Forms applications can access network resources.
D. Console-based applications do not display a graphical interface.

Answer : **D**

My Explanation: Windows form applications make use of a graphical interface that the user can interact with in order to complete their task. A console-based application is simple entry-response based code in which a user enters information and are given an output in a console-style that lacks a graphical interface.

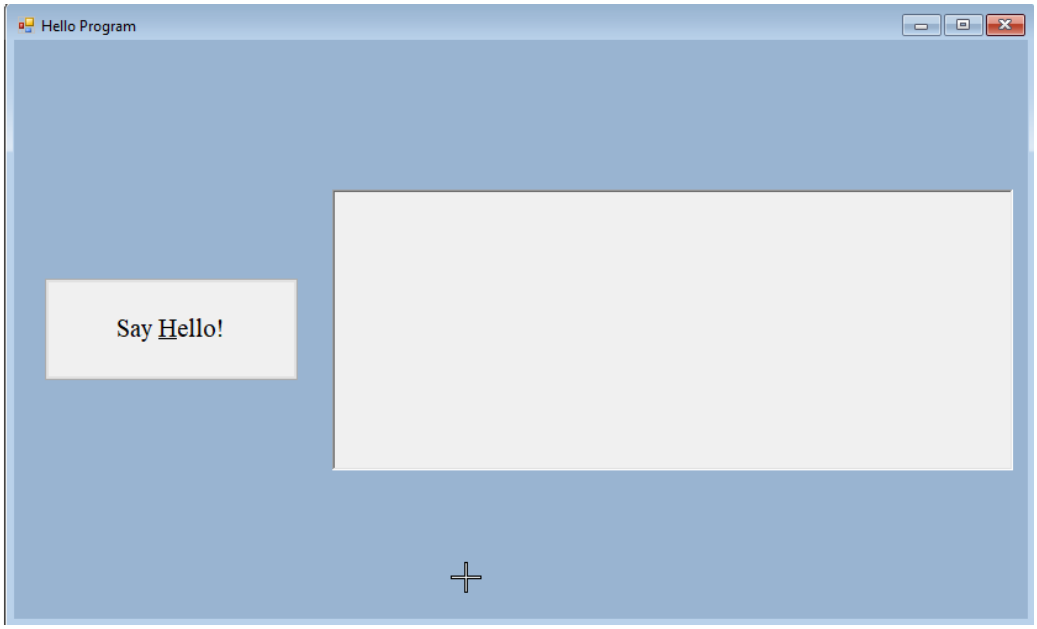
Ex. Of a console-based application

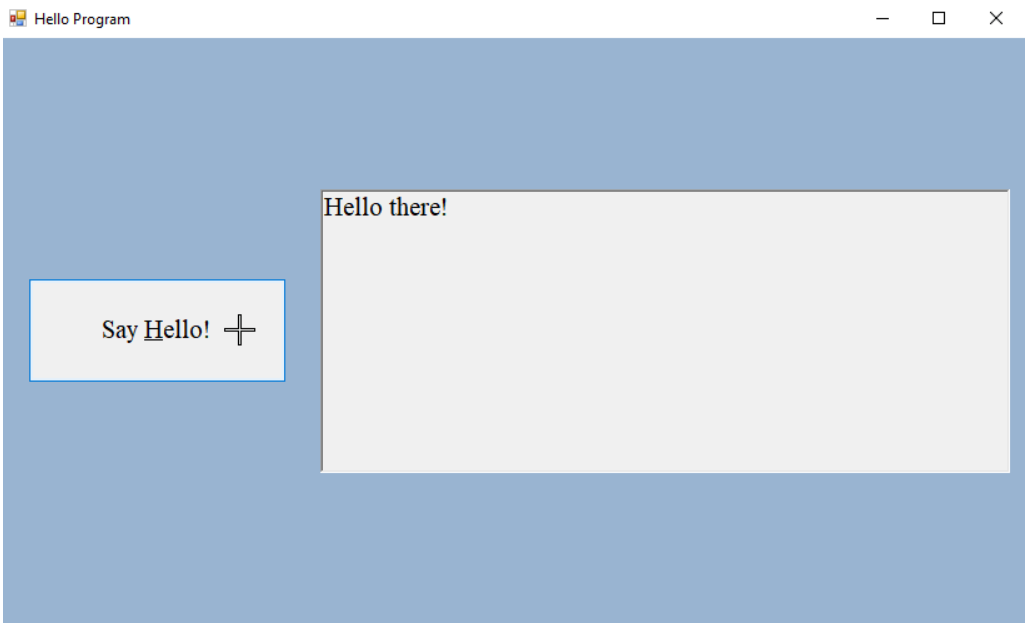
```
1  Module Module1
2
3      Sub Main()
4          Console.WriteLine("Hello World!") 'Writes "Hello World" to the console
5          Console.ReadLine() 'Waits for a response and closes the program
6      End Sub
7
8  End Module
```

```
Hello World!
```

Ex. Of a Windows forms application

```
1  Public Class Form1
2      'When the "Say Hello" Button is pressed, then:
3      Private Sub HelloButton_Click(sender As Object, e As EventArgs) Handles HelloButton.Click
4          ResultsBox.Visible = True 'Renders the answer box as visible
5          ResultsBox.Text = "Hello there!" 'Adds "Hello there!" to the results box
6      End Sub
7  End Class
```


Hello Program
Say Hello!


Hello Program
Hello there!
Say Hello!

Question 43:

You need to evaluate the following expression:
(A>B) AND (C<D)
What is the value of this expression if A=3, B=4, C=4, and D=5?

A. 0
B. 4
C. 5
D. False
E. Null
F. True

Answer : **D**

Explanation: A>B is false.

My Explanation: Booleans are simply true/false calculations based on the presented information. In this instance, we are looking at two comparisons joined with an AND. If A>B and C<D, then the Boolean is True. If either or both are false, the Boolean is false.

Ex. Of a Boolean Result Checker

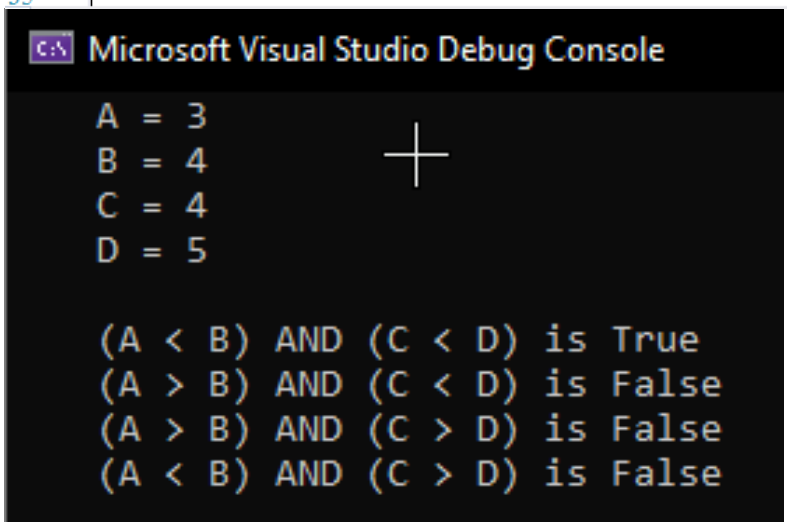
```
Imports System
Module Program
    Sub Main(args As String())
        'Declare your variables
        Dim A As Integer = 3
        Dim B As Integer = 4
        Dim C As Integer = 4
        Dim D As Integer = 5

        'Write the variables to the console
        Console.WriteLine("    A = 3" + vbCrLf +
                          "    B = 4" + vbCrLf +
                          "    C = 4" + vbCrLf +
                          "    D = 5" + vbCrLf)

        'Declare the Booleans for string format
        Dim Boolean1 As String = $"(A < B) AND (C < D)"
        Dim Boolean2 As String = $"(A > B) AND (C < D)"
        Dim Boolean3 As String = $"(A > B) AND (C > D)"
        Dim Boolean4 As String = $"(A < B) AND (C > D)"

        'Declare the booleans to check
        Dim CheckBoolean1 As Boolean = ((A < B) And (C < D))
        Dim CheckBoolean2 As Boolean = ((A > B) And (C < D))
        Dim CheckBoolean3 As Boolean = ((A > B) And (C > D))
        Dim CheckBoolean4 As Boolean = ((A < B) And (C > D))

        'Write the boolean strings as well as the checked true/false response.
        Console.WriteLine($"    {Boolean1} is {CheckBoolean1}" + vbCrLf +
                          $"    {Boolean2} is {CheckBoolean2}" + vbCrLf +
                          $"    {Boolean3} is {CheckBoolean3}" + vbCrLf +
                          $"    {Boolean4} is {CheckBoolean4}" + vbCrLf)
    End Sub
End Module
```



Question 107:

The purpose of the Finally section in an exception handler is to:

A. Execute code regardless of whether an exception is thrown.
B. Conclude the execution of the application.
C. Execute code only when an exception is thrown.
D. Break out of the error handler.

Answer : **A**

Website Explanation: *By using a finally block, you can clean up any resources that are allocated in a try block, and you can run code even if an exception occurs in the try block. Typically, the statements of a finally block run when control leaves a try statement. The transfer of control can occur as a result of normal execution, of execution of a break, continue, goto, or return statement, or of propagation of an exception out of the try statement.*

My Explanation: The finally block allows for users to have a block of code executed even if the rest of the "try" statement fails. The finally block will execute prior to the "try" statement exiting, no matter the reason of the exit.

Ex. Of a finally block

```
1   Imports System
2   Module Module1
3       Sub Main(args As String())
4           Try
5               Do
6                   Try
7                       'Tells the user to enter a number
8                       Console.WriteLine("Please enter a number")
9
10                      'Catches the input of the user, throws an error if the user doesn't enter an integer.
11                      Dim Number As Integer = Console.ReadLine()
12
13                      'Tells the user the number they entered
14                      Console.WriteLine($"The number you entered is {Number}!" + vbCrLf)
15
16                      Exit Do 'Continues Past The Do/Loop While
17
18                      'Catches an exception, such as if the user DID NOT enter a number
19                  Catch ex As Exception
20
21                      'Writes the error message to the console for the user to know what they did wrong.
22                      Console.WriteLine(ex.Message + Environment.NewLine)
23
24                  End Try 'Exits try
25
26              Loop While True 'Loops the above try sequence until it successfully reaches "Exit Do"
27          Finally 'Once the above is succesful, the program "finally" does this before moving forward.
28              'Thanks the user
29              Console.WriteLine("Thank you for using this program. Your testing means a lot to me." +
30                                Environment.NewLine)
31
32          End Try
33
34          Console.WriteLine(Environment.NewLine)
35      End Sub
36  End Module
```

```
Microsoft Visual Studio Debug Console
Please enter a number
a
Conversion from string "a" to type 'Integer' is not valid.

Please enter a number
1
The number you entered is 1!

Thank you for using this program. Your testing means a lot to me.
```

Question 113:

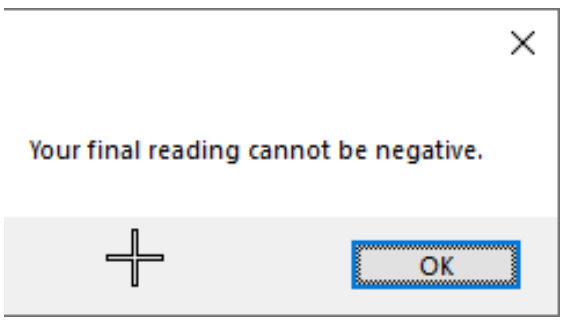
The throw keyword is used to perform which two actions? (Choose two.)

A. stop processing of the code
B. move error handling to a separate thread
C. raise exceptions
D. re-throw exceptions as a different type

Answer : **C,D**

Website Explanation: *Throw statement throws an exception that you can handle with structured exception-handling code (Try...Catch...Finally) or unstructured exception- handling code (On Error GoTo). You can use the Throw statement to trap errors within your code because Visual Basic moves up the call stack until it finds the appropriate exception- handling code.*
** This example throws an ApplicationException exception.*
Throw New ApplicationException -

My Explanation: The throw statement creates an exception, which you can create and adjust to your needs, and create an error-handling or exception-handling block to trap errors, handling the code properly rather than simply exiting and highlighting where the error occurred in your code.

Ex. Of throw statement

```
'Error handling - 'Alert/Display if Final is negative // Exit
If final < 0 Then 'Detects if entry is negative
    'Creates a new exception with a custom error message of what went wrong
    Throw New System.Exception("Your final reading cannot be negative.")
End If 'Continues the program
```