1ST Class – Tuesday evening, October 14, 2014

Instructor: Chris Oson – [coson@coleman.edu](mailto:coson@coleman.edu)

Assignments not on WebClass

Programming Assignment 1 due Nov 4th (3 weeks) (15 points)

Programming Assignment 2 (20 points)

Programming Assignment 3 (15 points)

Midterm (25 points)

Final (25 points)

Submitting assignments: Filename format: *COM350-xxxxxx-yy.zip*, 7z, rar, tar (make sure it compiles)

.NET Framework = DataType= Type

- Methods (0 or more methods)

- Properties (0 or more properties)

The ‘Object’ class is the queen bee. It has primitive types: integer, long, double …

We are adding 3 objects, not adding 3 integers

.NET Framework 1.0 – 2002: *C++, C#, J#, VB.NET*

.NET Framework 2.0 – 2005: *VB.NET (uses generics) ArrayList {a collection}*

LINQpad 4 –

Sub Main

Dim someVariable As Integer = 5: anotherVar = 15: yetAnotherVar = 25

‘ int someVariable = 5; anotherVar = 15; yetAnotherVar = 25;

‘ The add method has the following signature:

‘ fn Add(obj)

Al.Add(5)

Al.Add(10)

Al.Add(15)

Al.Add(5)

‘Al.Add(New DataTime(2007, 4, 15)

For Each item As Object in al

If item.GetType() = System.Int32 Then

‘Console.Writeline (item)

Else If item.GetType() = System.DateTime Then

‘Console.Writeline (item)

End If

Next

End Sub

‘ Flaw: not a type safe object, Do not use.

‘All primitive DataTypes reside on the stack.

‘Passing a value type as a reference type. An object, array, and a string - live on the heap.

Converting from a value type to a reference type - called ‘boxing’. It is a performance hit.

Converting back to a value type from a reference type - called ‘unboxing’. It is a performance hit.

List of T (in VB.NET):

List(of T) – it can be any datatype.

Dim myList As List(Of Integer) = New List(Of Integer)()

myList.Add(5)

myList.Add(10)

myList.Add(15)

For Each Val as Integer in myList

‘Do something

Next

- Uses lots of parenthesis.

In .NET, to determine the number of items in an array, use ‘length’ property.

But in a collection, like ArrayList, use the ‘count’ property

Use of Generics – Hand-in-hand with LINQ.

After break:

Cover core concepts of Assignment 1.

Create a pseudo Playlist stored on your laptop.

The idea is to WinForm Application

Launch ‘Option’ form with a FolderBrowserDialog1 control

Length of play list

Gap duration

Random(120-900) – (2 mins-15mins)

X amount of songs in the playlist as long as time is under the limit.

Gap duration will have x-1 occurrences.

In Python – I use a test-driven development approach.

We’ll use the test-driven development approach here too.

Waterfall development has evolved into ‘Agile’ development

Agile development – Sprints (1-4 weeks), 5-7 developers.

Every sprint, we ask what needs to be done.

- The programmers write the unit test and turn it into code down the line.

Our Unit Test:

0. Ensure the directory is valid.

1. Check if there are unit test waiting.

2. Look at business requirements.

Break it down:

1. A directory must have files (music files)

2. Gap duration must be greater than zero.

3. My playlist must be greater than 20:00

4. Every song must have min/secs.

5. Generate a random number (between120 to 900) for the duration.

6. Convert the number into a minute and seconds format

7. Aggregate the song times for the playlist length

8. If we clear the playlist it must reset the counter back to zero.

9. The gap number is equal to the song count – 1

10. Buttons are active as needed.

11. Empty playlist

Select “Show all files” to add a nunit.framework which creates the folder ‘reference’

Go to nunit.org and download 2.6 zip nunitframework.dll

Brand new project

Select Class Library – NewAssignment1Test to make ‘class1.vb’

- Import nunit

[www.skoolrox.com/COM350/unit-test.html](http://www.skoolrox.com/COM350/unit-test.html)

Allow the user to enter the full-qualified file path.

Show all files in Solution Explorer

Created folder DLLs and copy the nunit to it.

\_ ‘underscore’ is the continuation character.

Add attribute <TestFixture> \_

Imports System.IO

Imports Nunit.Framework

<TestFixture> \_

Public Class Assignment1Test

<Test> \_

Public Sub Does\_The\_Directory\_Exists()

‘Arrange

Dim testFolder As String = “D:\TEMP\Starting Location”

…

‘Act

Dim isValidFolder = Directory.Exists(testFolder)

…

‘Assert

Assert.True(isValidFolder, “The folder should exist.”

End Sub

<Test> \_

Public Sub Does\_The\_Directory\_DoesntExist()

‘Arrange

Dim testFolder As String = “D:\TEMP\Starting Location Whatever”

…

‘Act

Dim isValidFolder = Directory.Exists(testFolder)

…

‘Assert

‘Assert.False(isValidFolder, “The folder should not exist.”

Assert.False(ValidateFolderLocation(testFolder), “The folder should not exist.”

End Sub

Private Function ValidateFolderLocation(ByVal location As String) As Boolean

Return Directory.Exists(Location)

End Function

End Class

A subroutine that returns nothing – void

All unit tests – 3A’s: Arrange, Act, Assert.

Arrange – Can include a variable assignment

Act –

Assert –

Test runners

MS made a visual test runner

Compile your class

Can use the nuget to get the nunit.

To Refactor – create a method ValidateFolderLocation

Msdn – running unit test with explorer

Test – windows – test explorer

Tools- Library Package Manager – Package Manager Console

.vsx file – visual studio gallery file

Note: Nuget not always displays correct install status after a successful install.

You can Google “visual studio gallery nunit test adapter”

Next, we want to generate a random number.

Is it really random?

<Test> \_

Public Sub

‘Arrange

Dim rnd As Random = New Random()

‘ …

‘Act

Dim firstNumber As Integer = rnd.Next(120, 900)

Dim secondNumber As Integer = rnd.Next(120, 900)

‘…

‘Assert

Asset.That(firstNumber = secondNumber, “The numbers should not be equal.”)

End Sub

Create another function to give back a number. In real world we may want to pass in parameters.

In VB, we do Integer division with a forward slash: “\”

In VB, we do modulus operator with “MOD”

Or right-click on project name and select droplost item – Manage nuget packages – It goes online.

PM> Install-Package NUnit

2nd Class – Tuesday evening, October 21, 2014

Data source.

Word counting test

This is a Test, test, test,

Test -3

This -1

is -1

A -1

New Project – Installed – Templates – Visual Basic – Test – Unit Test Project

“WordCountTest.vbproj”

ListArray has ReadAllLines

Imports Microsoft.VisualStudio.TestTools

Private filePath As String

<TestClass()> Public Class

<TestInitialize()> \_

‘Dim filePath As String = “D:\TEMP\Sydney.txt”

Dim isValidPath = File.Exists(filePath)

<TestMethod()> \_

Public Sub The\_Container\_Has\_Data()

Dim allLines() As String = File.ReadAllLines(filepath)

Assert.IsTrue(allLines.Length > 0, filepath & “ should have data.”)

End Sub

End Class

‘ Tokenize it by the blank space and give an array.

‘Must see if the word has punctuation so we need to check the last symbol of the word and is it a punctuation mark.

<TestMethod()> \_

Public Sub Does\_The\_Token\_End\_With\_A\_Symbol()

token As String = “Hello,”. Trim() ‘ all leading and trailing white space

Dim lastCharacter As String = token.Substring(token.Length - 1)

Dim symbols() As String = {“;”, “,”, “’”, “.”}

Dim found As Boolean = False

Dim index As Integer = 0

End Sub

While index < token.Length and Not found

If symbols(index) = lastCharachter Then

Found = True

End If

Index += 1

End While

Assert.IsTrue(found, “ The last character should be a symbol.”)

End Class

‘Refactor cause we’ll use it and create a new method, not decorate it.

Private Function Sub DoesTheTokenEndWithASymbol(token As String) As Boolean

Dim lastCharacter As String = token.Substring(token.Length - 1)

Dim symbols() As String = {“;”, “,”, “’”, “.”}

Dim found As Boolean = False

Dim index As Integer = 0

While index < token.Length and Not found

If symbols(index) = lastCharachter Then

Found = True

End If

Index += 1

End While

Assert.IsTrue(found, “ The last character should be a symbol.”)

‘Perl has two functions: chop

<TestMethod()> \_

Public Sub Chop\_String()

Dim truncated As String = input.Substring(0, input.Length - 1)

Assert.IsTrue (truncated = “Hello Worl”, “The two values should be the same”) ‘ Conditional test

End Sub

<TestMethod> \_

Public Sub Chop\_String()

Dim truncated As String = chop(“Hello World”)

Assert.IsTrue (truncated = “Hello Worl”, “The two values should be the same”) ‘ Conditional test

End Sub

<TestMethod> \_

Public Sub Process\_Token\_With\_Symbol()

Dim line As String = “No, Sam I am. I do not like green eggs and ham.”)

Dim tokens() As String = line.split(“ “c)

Dim output As String = String.Empty

If DoesTokenWithASymbol(tokens(0)) Then

Output = Chop(tokens(0))

End If

Assert.AreEqual(“No”, output, “The items should be the same”)

End Sub

“Free it ebooks”

“Request response model”

The url is a virtual address. Convert it to a physical address.

Response: Header (mime type, status code) and the payload

Mime type is a mapping to a particular type. Example “text/html”

Next mod COM310 and the ASP.

We want to use the VB Dictionary class that takes parameters: T and value

String is word, integer is the count

Only the key gets the (Of before it.

Inferring a variable type by its declaration

A new instance

Accessor, the name of the variable, yhe unit type

Use the Dim command to allocate data for you.

In VB 2010 it is not necessary to declare the data type.

Dictionary(Of T, T)

Dim words As Dictionary(Of String, Integer) = new Dictionary(Of String, Integer)

Dim words = new Dictionary(Of String, Integer)()

First, I look for the token in the dictionary and if it is there I assign it an Integer of 1. The logic is:

If the key exist

Increase the count of the key

Else

Initialize the count to 1

IEnumerable(Of T)

We want to search through an array and stop when it is found.

If we had an array of strings:

Dim found = False

Dim tokens = {“one”,“two”,“three”,“four”,“five”,“six”,“seven”}

Dim index = 0

While index < tokens.Length and Not Found

If tokens(index) = “three” Then

Found = True

End If

Index = index + 1

End While

Dim tokens = {1,4,21,17,36,82}

The any method

Dim results = tokens.Any(Function(t) t >20)

Dim fileExts = {“mp3, “wma”, “ogg”}

Dim musicFiles = Directory.GetFiles(path)

Dim areValidFiles = From m in musicFiles

Let fi = new FileInfo(m)

Where fileExts.Any(Function(fi) fi.Extension)

3rd Class – Tuesday evening, October 28, 2014

CONTROL TREE

Form (Load)

- Label

- Button

- ListBox

List Control

- ListBox, ComboBox

Primitive data types – Min value, max values

Int32, 0

Decimal, 0.0

Double,

Int64,

Boolean,

DateTime, 1/1/0001

Char, 0

.Net 1.0

Convert.ToXXXX {Primitive data types, listed above}

Casts only primitive types

.Net 2.0

TryParse – It doesn’t throw the exception, it handles it gracefully.

Returns a Boolean value

DirectCast works with custom data types.

You have to declare a class with some properties.

To define classes in VB.NET we do:

Create a private field,

Create a public property

‘Auto Property’ equates to the setters and getters

Example:

Public Property Title As String

Keywords:

ReadOnly (no setter)

WriteOnly (no getter)

Example:

Public ReadOnly Property Title As String

Intellisense with the keyword *“Property”*

Type “prop” then TAB, then TAB. (the Intellisense kicks in)

It creates all the code for me.

*Create:*

Dim daysOfWeek() As TheDayOfWeek = { \_

New TheDayOfWeek With { .number = 1, .dayOfWeek = “Sunday”}, \_

New TheDayOfWeek With { .number = 2, .dayOfWeek = “Monday”} \_

}

Public Structure TheDayOfWeek

Dim number As Integer

Dim dayOfWeek As String

End Structure

Classes

- Constructors

- Properties / Fields

11/04/2014

- Finish classes

- Go over assignment #2

- Additional controls

Sub Main

’94 minutes and 25 seconds

Dim inputString As String = “94:25”

Console.Writeline(ValidateMinutes(inputString)

End Sub

‘Define other methods and classes here

Public Function ValidateMinutes(input As String) As Integer

Dim trimmedInput As String = input.Trim()

If trimmedInput.IndexOf(“:”c) = -1 Then

Return -1

End If

Dim tokens = trimmedInput.Split(“:”c)

Dim results As Integer = 0

If Int32.TryParse(tokens(0), results) = false Then

Return -1

End If

If Int32.TryParse(tokens(1), results) = false Or tokens(1).Length = 1 Then

Return -1

End If

If results < 0 Or results > 59 then

Return -1

End If

Return Results

End Funtion

4th Class – Tuesday evening, November 04, 2014

Application/Program

- Composed of one/more classes

- Classes are self contained

- 0+ more properties

- 1+ more method

- When a class is modified, it may or may not affect the other classes in a program.

“Pattern Searcher Class”

COMXXX-YYYYY-ZZ.zip, (z: 2a, 2b, 2c)

1. Traverses a directory for one or more files depending on the file type.

a. The user \*must\* provide a valid file type to execute the search

b. The following extensions are the valid file types:

- \*.\* (is executed to \*.txt, \*.html, and \*.xml)

(Note: \*.\* can be found in a binary file)

- \*.txt

- \*.html

- \*.xml

2. Once the list of files have been assembled based on one or more file types.

3. Iterate through the list of files searching for a pattern on each line.

a. The search may (or may not be case sensitive).

b. If the user does not provide a pattern, do not execute the search.

4. The pattern may or may not exist on each line.

a. The pattern may exists multiple times on one line.

5. Create an aggregate count of how many found results were encountered.

a. Create a type (SearchResults) that contains metadata about the found results.

b. The structure of the FoundResults should be:

i. Time – in milliseconds it took to locate the results.

ii. Matches – How many matches were found in one or more files.

iii. List(Of FoundResults) – see below

c. In addition to the aggregate count, use Generics to produce aList (Of FoundResults)

d. The structure of FoundResults should be:

i. Base file name

ii. Line number

iii. The input line that contains the pattern

Example output: (C:\temp\files\\* files)

“html”

Base file name, line no, input line that contains the pattern

First.html – 1: <html>

First.html – 78: </html>

Second.html – 2: <html>

Second.html – 1788: </html>

Third.html – 42: <html>

Third.html – 42: </html>

.NET Framework 4.5, Visual Basic – Class Library – “Driver Classes”

Add new item – Class Diagram

PatternSearcher

1. Create class “FoundResults” (with properties: BaseFileName, InputLine, LineNumber)

2. Create class “SearchResults” (with properties: Results-is aList(Of Foundresults), Time)

3. Create class “PatternSearcher” (with properties: location, Time)

Lesson: (OOP)

“Container classes”

Person Class

+Gender

+Height

+DOB

Student Class

+Student Id

+Major

Employee

Three Part Project (20 Points)

11-11-2014 – Unit Tests are due (7 out of 20) – 8 Points

11-20-2014 – The classes are due – 8 Points

11-18-2014 – Specs given

11-25-2014 – Assignment 2- Part III is due (Specs given on 11/18) – 4 Points

Can be 75-150 lines of code.

Module – Shared Class (it’s static)

Private FunctionGetAge(dob As DateTime) As Integer

Dim ts As TimeSpan = DateTime.Now – dob

Dim dayCount As Integer = ts.Days

Dim totalDays As Decimal = Convert.ToDecimal(dayCount / 365.25)

Return Convert.Int32(Math.Floor(totalDays))

End Function

Public Sub Class Main

‘Object Initializer

Dim onePerson As Person = New Person With { \_

.Gender = “F”c, \_

.Height = 61, \_

.Dob = New DateTime(1977, 11, 14) \_

}

Dim age As Integer = onePerson.Age

Dim anotherPerson As Person = New Person With { \_

.Gender = “M”c, \_

.Height = 68, \_

.Dob = New DateTime(1984, 3, 14) \_

}

Dim personCollection As List(Of Person) = New List (Of Person) ()

personCollection.Add(onePerson)

personCollection.Add(anotherPerson)

5th Class – Tuesday evening, November 11, 2014 – No Class – Veterans’ Day Holiday

6th Class – Tuesday evening, November 18, 2014

Expect to have one or more constructor(s)

Diagram shows four properties

Domain Transfer Objects (DTOs)

Tonight’s Topics:

Interviewers asks:

- OOP. What is an abstract class? – Answer: It is a class that must be inherited from;

- you cannot instantiate an object of it directly. You can its derived members.

- it may or may not have abstract methods and/or abstract properties

- its implementation can be incomplete

Example: Stream Class... This is an abstract class.

Derived members (subtypes) of the Stream Class:

- BufferedStream

- NetworkStream

- CryptoStream

- FileStream

- MemoryStream – handles large files and subdirectories

- - - -

<https://social.msdn.microsoft.com/Forums/vstudio/en-US/93d1d97b-b8cb-4026-bf36-a28db936a3cc/in-order-to-debug-this-project-add-an-executable-project-to-this-solution-which-references-this?forum=vbgeneral>

Question: How do I reference the library project when I get this error?

"A project with an output type of class library cannot be started directly. In Order to debug this project, add an executable project to this solution which references this library project.  Set the executable project as the startup project."

Answer: In your IDE, go to File\Add\New Project and   
1) choose WindowsApplication [or ConsoleApplication].   
2) In the Solution Explorer right click on the WindowsApplication project and choose "Set  As Start Up".   
3) Right click on it again and choose "Add Reference".   
4) Click the Projects Tab and Select your class library project.   
5) You can now reference your class library in code from the windows application like so:  
    Private Sub Button1\_Click(ByVal sender As System.Object, \_  
 ByVal e As System.EventArgs) Handles Button1.Click  
        Dim tc As New ClassLibrary1.Class1  
        Dim Sum As Integer = tc.Add(5, 2)  
        MsgBox(Sum.ToString)  
    End Sub

Well, it turns out that all you need to enable debugging in Visual Basic Express Edition is a separate file in the project folder, called "MyProjectName.vbproj.user". This contains the user-specific project settings for the MyProjectName project. In my test I used the default project name for a Class Library, and so the file was called "ClassLibrary1.vbproj.user" and in the same folder as "ClassLibrary1.vbproj".

Here is what the file needs to read:   
<Project xmlns="<http://schemas.microsoft.com/developer/msbuild/2003>">  
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">  
    <StartAction>Program</StartAction>  
    <StartProgram>C:\Program Files\Your Application\Application.exe</StartProgram>  
  </PropertyGroup>  
</Project>

After creating this file (please make sure the executable path is correct for your system, of course), you will need to reopen the project for it to be detected and read. But once it's found, you can debug your class library project with an external executable, just by hitting F5.

Sub Main

‘To create a class and consume this class

Dim tc As TangentCalculator = New TangentCalculator()

Console.Writeline(tc.Add(5, 15))

End Sub

Public Class TangentCalculator : **Inherits** Calculator

‘ Colon : is an alias for the new line character

Public Sub New()

Public **Overrides** Function Multiply(first As Integer, second As Integer) As Integer

Return first \* second

End Function

Public Overrides Function Divide(first As Integer, second As Integer) As Integer

‘IIF is VBs Tiernary operator

Dim results As Double = IIf(second = 0, \_

Convert.ToDecimal(0), \_

Convert.ToDecimal( first / second ))

Return results

End Function

End Class

‘Define other methods and classes here

Public **MustInherit** Class Calculator

‘Native methods – cannot be overridden.

‘The derived type cannot modify or override the Add method in the abstract class

Public Function Add(first As Integer, second As Integer) As Integer

Return first + second

End Function

Public Function Subtract(first As Integer, second As Integer) As Integer

Return first - second

End Function

‘This is the abstract method because of the Mustoverride keyword

‘and the method definition is empty

Public **MustOverride** Function Multiply(first As Integer, second As Integer) As Integer

‘ Third type is an optional method using the Overridable keyword

‘Abstract method \*CAN\* be overridden in the derived type. If not, then use this implementation

Public **Overridable** Function Divide(first As Integer, second As Integer) As Double

If second = 0 Then Return Convert.ToDouble(0)

Return Convert.ToDouble(first / second)

End Function

End Class

Project Note:

If you're using an external dll then in your VS solution, create a dll folder, right-click folder

In a text file state “My project has a dependency in the DLLs folder.”

Abstract class:

Public MustInherit Class Mammal

Public Property Gender As Char

End Class

Public Class Person : Inherits Mammal

Private canSetDateOfBirth As Boolean

Private \_dob As DateTime

Public WriteOnly Property Dob() As DateTime

Set(ByVal value As DateTime)

If canSetDateOfBirth Then

\_dob = value

\_age = ComputeAge(\_dob)

canSetDateOfBirth = False

End If

End Set

End Property

Private \_age As Integer

Public ReadOnly Property Age() As Integer

Get

Return \_age

End Get

End Property

Private Function ComputeAge(birthDate As DateTime) As integer

Dim ts As TimeSpan = DateTime.Now - birthDate

Dim years As Double = Convert.ToDouble(ts.Days / 365.25)

Dim roundedYears As Integer = Convert.ToInt32(Math.Floor(years))

Return roundedYears

End Function

Public Sub New()

canSetDateOfBirth = True

End Sub

End Class

‘TimeSpan Class. We are interested in properties: Days

Dim ts As TimeSpan = DateTime.Now - \_dob

Dim days As Integer = ts.Days

Dim year As Integer = 0

Dim results = Int32.TryParse(Days / 365.25)

Vb.net, C3, F# - MSIL – exe

Module is the equivalent of a shared class. Use ‘Show all File’ and ‘Collapse All’ – Add References – Add Abstract Examples – Solution –

Imports AbstractExamples

Module TestRunner

Sub Main

Dim p As Person = New Person()

p.Gender = “F”c

p.Dob = New DateTime(1969, 7, 15)

Dim mask As String = “The persons age is: {0}”

Console.Writeline(mask, p.Age)

Pause()

End Sub

Public Sub Pause(Optional ByVal message As String = \_

“Press any key to continue”)

Console.Writeline(message)

Console.ReadKey()

End Module

Question: How do object-oriented programming languages implement polymorphism?

‘Polymorphism’ is to take on many shapes.

Answer: Polymorphism is implemented through inheritance.

Optional keyword as an optional parameter with a default data type and it must always be last.

If an argument is passed then the optional parameter is overridden.

Parameters.Add

Parameters.AddRange

Parameters.AddWithValue

Leave and Enter

General syntax for stored procedure in MySql:

Fully-qualified name of the object (database.objectname):

CREATE PROCEDURE `COM350`.`Languages`

(

IN ProgLanguage VARCHAR(32)

)

BEGIN

(When PHP exports data to a SQL script it puts `backward-ticks` in the object names)

General syntax for stored procedure in SqlServer:

7th Class – Tuesday evening, November 25, 2014

[www.omnicon-innovations.com/snippets/SQL/MySQL](http://www.omnicon-innovations.com/snippets/SQL/MySQL)

Create a new folder DLL and put the new drivers in there.

After you download the project you still have to import the assemblies yourself.

The references to the assemblies and import in the code too.

[www.connectionstring.com](http://www.connectionstring.com)

Google: Repository Interface

Dummy method… using … implements IDisposable

Abstract classes and interfaces.

What is an interface?

“An interface is a contract that defines the signature of some piece of functionality. This contract is fulfilled by the implementing of the class. The implementing class \*MUST\* execute “the contract” or all of the methods of the interface.”

Public Interface IMyInterface

‘Properties, Methods, and Events can live in the interface

‘Private fields and Accessors CANNOT reside in an interface

End Interface

---------

Solutions to project:

Public Class PatternSearcher

- 4 properties, 1 private field

Methods…

‘Load the files, getfiles

Private Sub BuildList()

Private fileList As String()

fileList = Directory.GetFiles(fileType)

End Sub

3rd Part of Assignment:

GUI on phone.

-----

8th Class – Tuesday evening, December 2, 2014

In-line SQL method – This is not a best practice because it is time consuming, tedious, prone to attacks.

Instead, use a mask.

Use ‘parameterized queries.’

Use the String format:

Sub Main

Dim first As Integer = 5: Dim second As Integer = 10

Dim third As Integer = first + second

Dim addition as String = String.Format(“{0} + {1} = {2}”, \_

First, second, third)

Console.WriteLine(addition)

Dim connectString As String = “CONNECTION STRING”

‘ VARCHAR (50) NOT NULL

Dim sqlQuery As String = “INSERT INTO Language (Language) VALUES ‘” & \_

“(@Language)”

Dim getLastLanguage As String = “SELECT TOP 1 LANGUAGE FROM Languages “ & \_

“ORDER BY Languageid DESC”

Dim language As String = string.empty

Dim rowsAffected As Integer = 0

Using sqlConnection As SqlConnection = New SqlConnection(connectionString)

Using sqlCmd As SqlCommand = New SqlCommand(sqlQuery, sqlConnection)

sqlConnection.Open()

sqlCmd.Parameters.Add(“Language”, SqlDbType.VarChar, 50).value = “Algol”

‘replaced: sqlCmd.ExecuteNonQuery() ‘insert, update, delete

rowsAffected = sqlCmd.ExecuteNonQuery() ‘insert, update, delete

sqlCmd.CommandText = getLastLanguage

language = sqlCmd.ExecuteScalar().ToString()

End Using

End Using

Console.WriteLine(rowsAffected)

Console.WriteLine(language)

End Sub

Output:

1

Algol

Everything is an object so if you are in the wrong database you will get an error “object not found”

Sql “limit 1” is the same s T-Sql “top 1”

For example: limit 41,10 ‘is the fifth set of ten records.

Stored Procedure

VBParameterimedQuery

T-SQL Transact SQL

- All variables start with the “@” symbol

- The variables names in parameterized queries are case-sensitive (EVEN-THOUGH T-SQL is not.)

Pay attention to the Columns {definitions}

There is a mapping between sql data types and .net data types.

Decimal (Precision, Scale)

1/1/1754 – 12/3/2099

Starts with an ‘N’ is Unicode. Different than the ASCII table (255 values). Unicode table ( values)

A-Z, a-z, 0-9,

ASC 32 – Space

ASC 9 – TAB

ASC 13/10 – CR/LF

<h2>I own 10% of the car</h2>

Research the ASCII table

&#60;h2&#62;No HTML&#60;&#63;h2&#62;

RTF Formatter

Olot..

u

En ~

Hex is made by dynamic URLs.

61 3D =

37 25 %

Sub Main

Dim first As Integer = 5: Dim second As Integer = 10

Dim third As Integer = first + second

Dim addition as String = String.Format(“{0} + {1} = {2}”, \_

First, second, third)

Console.WriteLine(addition)

Dim connectString As String = “CONNECTION STRING”

‘ VARCHAR (50) NOT NULL

Dim sqlQuery As String = “INSERT INTO Language (Language) VALUES ‘” & \_

“(@Language)”

Dim getLastLanguage As String = “SELECT TOP 1 LANGUAGE FROM Languages “ & \_

“ORDER BY Languageid DESC”

Dim language As String = string.empty

Dim rowsAffected As Integer = 0

Using sqlConnection As SqlConnection = New SqlConnection(connectionString)

Using sqlCmd As SqlCommand = New SqlCommand(sqlQuery, sqlConnection)

sqlConnection.Open()

sqlCmd.CommandType = Commandtype.StoredProcedure

sqlCmd.Parameters.Add(“Language”, SqlDbType.VarChar, 50).value = “Objective C”

‘replaced: sqlCmd.ExecuteNonQuery() ‘insert, update, delete

rowsAffected = sqlCmd.ExecuteNonQuery() ‘insert, update, delete

sqlCmd.CommandType = Commandtype.Text

sqlCmd.CommandText = getLastLanguage

language = sqlCmd.ExecuteScalar().ToString()

End Using

End Using

Console.WriteLine(rowsAffected)

Console.WriteLine(language)

End Sub

Output: -1, Objective C

Important – When you use a stored procedure you need to change the command type

commandType = text (default)

commandType = StoredProcedure

commandType = TableDirect

SET NOCOUNT ON

SET NOCOUNT OFF

ALTER PROCEDURE InsertNewLanguage

….

EXEC InsertNewLanguage

Switch gears completely and go over to MySQL

It uses the back tick character more than the double quote character.

Need:

1. Database name

2. Database owner dbo

3. Object

EXEC InsertNewLanguage ‘Expect’

-- Fully qualified name of any object is:

-- [DATABASE].[OWNER].[OBJECT]

[ColemanDevelopment].[dbo].[object]

SQL Server

-- Fully qualified name of any object is:

-- [Database\_name].[Owner].[Object]

MySQL

-- Fully qualified name of any object is:

-- `Database\_name`.`Owner`.`Object`

But…when you add your parameters you have to look up the MySQL parameters MySQL.DataString…

Sub Main

Dim connectString As String = “Server=localhost:3306;Database=Coleman”

Dim msqlConnection As New SqlConnection = New SqlConnection(connectionString)

Dim connectText = “D”

Dim mySQLConnection = File.ReadAllText(connectionText)

Dim connectionString As String = mySQLConnection

Using sqlConn As MySQLConnection = New MySQLConnection(connectionString)

sqlConn.Open()

Using sqlCmd As MySQLCommand = New MySQLCommand(sqlQuery, sqConn)

rowsAffected = sqlCmd.ExecuteNonQuery()

End Using

Console.WriteLine(“Rows affected: {0}”, rowsAffected)

End Using

sqlConnection.Open()

Console.WriteLine(“The database is open”)

sqlConnection.Close()

sqlCmd.Parameters.Add(“Language”, SqlDbType.VarChar, 50).value = “Algol”

End Sub

DataGridView control is the most powerful control for functionality.

Binding data to a DataGridView there are many options:

- Smart Tag [small icon in the corner]

In the Startup code

1) Retrieve the data from database.

2) IListSource

Stopwatch class – when you need 100x more accuracy

…than just subtracting DateTime.Now from earlier occurrence.

DIM sw As .start

.stop

Dim ts As TimeSpan(sw.elapsed)

9th Class – Tuesday evening, December 9, 2014

John O’Brien – How do you deal with the single quote.

Textbox (student name): ‘); DELETE QUERY;--

Dim sql = “INSERT INTO TABLE (StudentName)” & \_

“VALUES(‘ ” & sname.Text & “ ’)”

Here is the rendered text “‘); DELETE QUERY;--” after it’s been concatenated:

VALUES(‘’); DELETE QUERY;--‘)

Week 6 – Abstract classes

Week 7 – Interfaces

Week 8 – ADO.NET

Week 9 – ADO.NET

Week 10 – Exceptions, Final Exam