Assignment 5 – Utility Classes

This is a start to building a collection of utility classes … collections of methods that are small, focused, and applicable to multiple applications. You can put these classes into a Class Library project or into a folder in the Windows project used to test the classes.

Where you see ***XX***, insert your initials instead.

You will lose marks if you write redundant code … code that could be handled by one of the other methods.

Setup

1. Either create a new ***XXAssignment5*** Windows project or add an ***XXAssignment5*** Windows form to your multi-form project.
2. Add a folder called ***XXUtilityClasses*** to your project or add a Class Library project called ***XXUtilities*** to your solution.
    a. Create three <u>*public*</u> <u>*static*</u> classes in this folder (or Class Library):
        i. XXNumericUtilities
        ii. XXStringUtilities
        iii. XXValidations
    b. Get a clean build and, if you're using a Class Library, add a reference to it in your Windows project.

XXNumericUtilities Class

1. A number is defined as one or more digits with an optional leading dash and an optional single decimal place anywhere in the number.
    a. Create two Boolean methods called ***IsNumeric*** … one accepts a string parameter and the other accepts an Object parameter. They both return true if the parameter contains only a number.
2. An integer is defined as a number that does not contain a fractional portion (no decimal place).
    a. Create three Boolean methods called ***IsInteger*** … one accepts a string, one a double and one an object.
3. Users often enter currency symbols, commas, spaces and trailing dashes in numeric fields.
    a. Create a string method that returns the string less everything except digits, the decimal place and a dash. If there was a <u>single</u> dash (leading or trailing), put that as the first character. If the result is numeric, return it as a string, else return null.
4. In the *KeyPress* event handler for a textbox, the *KeyPressEventArgs* "e" has two properties:
    - e.KeyChar – the character the user is trying to add to the textbox's contents
    - e.Handled – if true, the character will not be added to the textbox's contents

    a. Create a void method that accepts a *KeyPressEventArgument* as a parameter. If the property *KeyChar* is not a digit, set the property *Handled* to true. The Intelligence light-bulb should help you add a reference to Windows.Forms.

XXStringUtilities

1. Users expect a little intelligence in apps these days: when typing names, street addresses, city names, etc., they expect the app to correct their typing.

   a. Write a static string method that accepts a string as a parameter. Have it drop the whole string to lower case, then capitalize the first letter of every word. Return the capitalized string. If the original string was null, return null.
2. Sometimes users use strange punctuation for numeric strings like phone numbers, credit card numbers, account numbers, zip codes, etc., which make comparisons difficult.
   a. Write a static string method that accepts a string parameter and returns a string with just the digits from it.
3. I like to standardise the format of things when I store them into a database:
   a. Write a static string method that accepts a string with 7 or 10 digits and reformats it into either the 123-1234 or 123-123-1234 pattern.
   b. Write a static string method that accepts a string representing a Canadian Postal Code, shifts it to upper case and inserts the space if it is missing (and the length permits doing so). If the original string is null, just return null.
   c. Write a static string method that accepts a string with 5 or 9 digits and reformats it into either the 12345 or 12345-1234 pattern.
4. Full name is useful for reports, comparisons and so on.
   a. Create a string method called ***FullName*** that accepts two strings (any case): one the first name, one the last name, and returns the full name, capitalized, in the form: *Turton, David*.
      i. If only one of the names is provided, just return that, with no punctuation.
      ii. If neither name is provided, return null

XXValidations

1. Write a Boolean method that validates a Canadian Postal Code pattern (A3A 3A3), ensuring the first letter is one of the valid 18 letters, and the remaining letters are one of the valid 20 letters. Accept either upper case or lower case and the space is optional.
   a. If the string is null or empty, return true.
2. Write a Boolean method that extracts the digits from a string representing a US Zip Code and confirms that it contains either 5 or 9 digits. This should be a one-liner.
3. Write a Boolean method that extracts the digits from a string representing a phone number and confirms that it contains exactly 10 digits.

Class Test Form

1. Create one form that tests all of the above methods and their overloads.
   a. It should have one ***submit*** button as the default "Accept" button for the form, executing all of the methods* at once.
   b. Each method should have one line, consisting of:
      i. a label naming the method being tested
      ii. an input textbox(es) for its parameter(s) (the string version of it)
      iii. a label showing the result from the method
2. The KeyPress event handler will exhibit its results as the user types into the textbox
3. * For overloaded methods expecting data types other than a string, their tests should convert the textbox input into the pertinent datatype before calling the method.

Hand In

- Hand in:
  - The standards marking sheet, with your name on it

- The marking sheet for this assignment, also with your name on it.
- Print your XXProvince and XXProvinceMaintenance code … did you put your name in the program comments?
- **Due: Wed Aug 8, 2018**