INFO 1112: Assignment–3

School of Business
Kwantlen Polytechnic University, Canada
`https://courses.kpu.ca/course/view.php?id=6371`

Instructor: Ali Madooei
`ali.madooei@kpu.ca`

June 24, 2016

It is due at 12:00 pm (noon) on Friday, July 1, 2016

This assignment is designed around the content of Chapters 6 [Gadis, 2014]: methods. It contains one task and carries 20 marks.

Guidelines

1. Create a C# project `Task01` in a C# solution called `Assign03`.
2. Attempt to implement the application and do your best. Just as there are different ways to say the same thing in English, there are different ways to do the same thing with a programming language. Any solution is acceptable as long as the code is error-free, implements what I have asked for, and produces the correct output.
3. I encourage you to follow good programming practice (e.g. include comments, use meaningful identifiers, code indentation, etc). This practice will be considered in marking.

Task 1. In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. The algorithm is useful to determine if a card number is entered correctly or if a credit card is scanned correctly by a scanner. Almost all credit card numbers are generated following this validity check, commonly known as the Luhn check or the Mod 10 check, which can be described as follows (for illustration, consider the card number 4388576018402626):

1. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the two digits to get a single-digit number.

```
2 * 2 = 4
2 * 2 = 4
4 * 2 = 8
1 * 2 = 2
6 * 2 = 12 (1 + 2 = 3)
5 * 2 = 10 (1 + 0 = 1)
8 * 2 = 16 (1 + 6 = 7)
4 * 2 = 8
```

2. Now add all single-digit numbers from Step 1.

```
4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37
```

3. Add all digits in the odd places from right to left in the card number.

```
6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38
```

4. Sum the results from Step 2 and Step 3.

```
37 + 38 = 75
```

5. If the result from Step 4 is divisible by 10, the card number is valid; otherwise, it is invalid. For example, the number 4388576018402626 is invalid, but the number 4388576018410707 is valid.

Create a GUI similar to Figure-1 and implement Luhn's algorithm to validate credit card numbers. **Notes:** (1) When user clicks on "Validate" button, a Message Box will be shown to present the validation result. (2) Although user may input information about the type of the credit card and the expiration date, we are going to ignore these information for simplicity. We will only focus on Luhn's algorithm, using the credit card number for validation. (3) The radio button caption is a combination of image and text. You are not required to reproduce the same effect; text is enough. I do encourage you to try to implement it by referring to your textbook or the resources over the Internet. (4) The GUI controls used for "Expiration date" are *ComboBox*es. The Combobox control is similar (in many ways) to the Listbox control, in particular that both controls have an `Item` property. Refer to your textbook for more information.

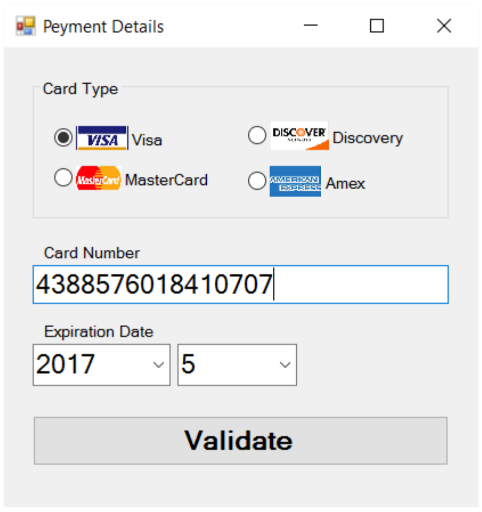

Figure 1: GUI for Credit Card validator application.

Important: You are required to implement the following methods and employ them in your application.

```
/** Return true if the card number is valid */
static bool IsValid(long number)
{
    /* Your code goes here */
}

/** Get the result from Step 2 */
static int SumOfDoubleEvenPlace (long number)
{
    /* Your code goes here */
}

/** Return this number if it is a single digit,
 * otherwise, return the sum of the two digits */
static int GetDigit(int number)
{
    /* Your code goes here */
}

/** Return sum of odd place digits in number */
static int SumOfOddPlace(long number)
{
    /* Your code goes here */
}
```

Optional: Credit card numbers follow certain patterns. A credit card number must have between 13 and 16 digits. It must start with:

- 4 for Visa cards.
- 5 for Master cards.
- 37 for American Express cards.
- 6 for Discover cards.

Feel free to enhance your application and consider theses patterns as well as the Luhn's algorithm.

Submission

Once you are done with your solutions, please include the following comment block (with you name and student number) on top of every C# file in your project.

```
/*
*  INFO 1112 - SUMMER 2016
*  ASSIGNMENT 3
*
*  Student Name:
*  Student Number:
*/
```

Then, zip the project `Assign03` folder and upload it to Moodle.