**Checking Credit Cards**

**Introduction**

Develop a solution to the below problem of how to **Validate Credit Cards.**

**Requirements**:

* Basic UI
* .Net Web API.
* Unit Tests.

**We are looking for:**

* Implementation of SOLID principals.
* Production ready code.
* Full test coverage.

There is provided sample data to be used for testing and to show examples of results.

**Problem**

Before submitting a credit card to a payment gateway it's important that we run some sanity checks on the number.

A common check that is performed upfront is to validate the card type based on the starting digits and length of card number. The main patterns that we care about are as follows:

+============+=============+===============+

| Card Type | Begins With | Number Length |

+============+=============+===============+

| AMEX | 34 or 37 | 15 |

+------------+-------------+---------------+

| Discover | 6011 | 16 |

+------------+-------------+---------------+

| MasterCard | 51-55 | 16 |

+------------+-------------+---------------+

| Visa | 4 | 13 or 16 |

+------------+-------------+---------------+

All of these card types also generate numbers such that they can be validated by the Luhn algorithm, so that's the second check systems usually try. The steps are:

1. Starting with the next to last digit and continuing with every other digit going back to the beginning of the card, double the digit
2. Sum all doubled and untouched digits in the number. For digits greater than 9 you will need to split them and sum the independently (i.e. "10", 1 + 0).
3. If that total is a multiple of 10, the number is valid.

For example, given the card number 4408 0412 3456 7893:

1 8 4 0 8 0 4 2 2 6 4 10 6 14 8 18 3

2 8+4+0+8+0+4+2+2+6+4+1+0+6+1+4+8+1+8+3 = 70

3 70 % 10 == 0

Thus that card is valid.

Let's try one more, 4417 1234 5678 9112:

1 8 4 2 7 2 2 6 4 10 6 14 8 18 1 2 2

2 8+4+2+7+2+2+6+4+1+0+6+1+4+8+1+8+1+2+2 = 69

3 69 % 10 != 0

This card is not valid.

**Input and Output**

Given the following credit cards:

4111111111111111

4111111111111

4012888888881881

378282246310005

6011111111111117

5105105105105100

5105 1051 0510 5106

9111111111111111

I would expect the following output:

VISA: 4111111111111111 (valid)

VISA: 4111111111111 (invalid)

VISA: 4012888888881881 (valid)

AMEX: 378282246310005 (valid)

Discover: 6011111111111117 (valid)

MasterCard: 5105105105105100 (valid)

MasterCard: 5105105105105106 (invalid)

Unknown: 9111111111111111 (invalid)