AP Computer Science A@Beijing National Day School

Problem Set 8: `Credit Card`

Due date: Monday, November 12, 2018
Instructor: Mr. Alwin Tareen
Total Points: 15

Task Overview

- Implement a program that determines whether a provided credit card number is valid, according to Luhn's algorithm.

Background

- Every credit card has a number, both printed on its face, and embedded in the magnetic stripe on its back. That number is also stored in a database somewhere, so that when your card is used to buy something, the creditor knows whom to bill.
- There are many people with credit cards in this world, so those numbers are pretty long:
  - American Express: 15-digit numbers
  - MasterCard: 16-digit numbers
  - Visa: 13- and 16-digit numbers
- Credit cards companies don't just assign a random series of digits to compose their credit card numbers. Rather, there is actually some structure to them:
  - American Express numbers must start with 34 or 37.
  - MasterCard numbers must start with 51, 52, 53, 54 or 55.
  - Visa numbers must start with 4.
- Credit card numbers also have a "checksum" built into them, which is a mathematical relationship between at least one number, and the others. This checksum enables computers to detect errors and fraudulent credit card numbers, without having to query a database, which can be slow.
- Credit card companies use an algorithm developed by Hans Peter Luhn, a researcher from IBM.
- According to **Luhn's algorithm**, you can determine if a credit card is valid by executing the following steps:
  1. Multiply every other digit by 2, starting with the number's **second-to-last digit**, and then add those products' digits together.
  2. Take this result, and add it to the sum of the digits that weren't multiplied by 2.
  3. If the total's last digit is 0, then that credit card number is valid.

- Consider an example of Luhn's algorithm with the following American Express number: 378282246310005

  1. For the sake of clarity, I have underlined every other digit, starting with the number's second-to-last digit:

     $$3\underline{7}8\underline{2}8\underline{2}2\underline{4}6\underline{3}1\underline{0}0\underline{0}5$$

     Then, multiply each of the underlined digits(highlighted in bold) by 2:

     $$\mathbf{7} \cdot 2 + \mathbf{2} \cdot 2 + \mathbf{2} \cdot 2 + \mathbf{4} \cdot 2 + \mathbf{3} \cdot 2 + \mathbf{0} \cdot 2 + \mathbf{0} \cdot 2$$

     The partial products are as follows:

     $$14 + 4 + 4 + 8 + 6 + 0 + 0$$

     Next, add those products' digits(*Note:* not the products themselves) together:

     $$1 + 4 + 4 + 4 + 8 + 6 + 0 + 0 \quad = \quad 27$$

  2. Now, add the result of 27 to the sum of the digits in the credit card number that *weren't* multiplied by 2:

     $$27 + 3 + 8 + 8 + 2 + 6 + 1 + 0 + 5 \quad = \quad 60$$

  3. Notice that the last digit in the result of 60 is a 0, so the credit card number is legitimate.

Specification

The Information Box Which Includes Your Name[5 points]

- Type your English and Pinyin name into the `Author` field, where it says: `YOUR NAME HERE`

Verify whether a Credit Card Number is Legitimate [10 points]

- Write a `Java` program in the file `CreditCard.java` that reports whether a credit card number is legitimate or not.
- You will write your solution in a function called `validate(String digits)`, right below the place where it says: `YOUR CODE HERE`.
- If the credit card number is valid, then your `validate` function should return a String corresponding to the specific credit card brand. These return values should be `AMEX`, `MASTERCARD`, `VISA`, or `VALID`. Otherwise, if the credit card number is not valid, then your `validate` function should return the String `INVALID`.
- Make sure that you run your `Java` program, and ensure that it is free of errors. The program includes a function call to `validate(String digits)`, in which the provided argument has a value of `"378282246310005"`. Therefore, the output of this program should be: `AMEX`

Hints

- Note that the credit card number is being brought into the function as a String. This means that you will have to use the `substring()` method throughout your program, to separate out the individual digits.
- You will have to convert between String values and integers. Use the `Integer.parseInt()` function to perform this conversion, as follows:

```
String sample = "123";
int num = Integer.parseInt(sample);
```

- Luhn's algorithm clearly specifies that you must sum the digits starting from the **second-to-last digit**. Does this mean that you have to perform a reverse loop? Not necessarily. Think about what happens when a credit card has an **odd** quantity of digits, it means that we have to sum all of the elements located at the **odd-numbered** indexes:

$$3\underline{7}8\underline{2}8\underline{2}2\underline{4}6\underline{3}1\underline{0}0\underline{0}5$$

- Now, think about what happens when a credit card has an **even** quantity of digits. In this case, we have to sum all of the elements located at the **even-numbered** indexes:

$$\underline{5}1\underline{0}5\underline{1}0\underline{5}1\underline{0}5\underline{1}0\underline{5}1\underline{0}0$$

- There are many different ways to solve this problem set, and you are free to choose any implementation strategy that you wish. However, the following is some pseudocode that outlines one particular strategy:

```
determine which start index to begin from:
    if credit card number length is even, begin from index 0.
    otherwise, if number length is odd, begin from index 1.
multiply every other digit by 2, then add the products' digits
sum the digits that weren't multiplied by 2
get the first two digits of the credit card number
determine the brand of the credit card number
return the credit card brand
```

Testing

- The file `CreditCardJUnitTest.java` contains the `JUnit` test cases which verify the correct functionality of the program.

Submission

- Submit your `Java` program by uploading it to the `Web-CAT` automated grading platform. Click on the following link:
  `http://ec2-54-65-207-33.ap-northeast-1.compute.amazonaws.com:8080/Web-CAT/WebObjects/Web-CAT.woa`