CS101- Algorithms and Programming I

Lab 06

Lab Objectives: `static methods`

- For all labs in CS 101, your solutions must conform to the CS101 style guidelines (rules!)

  Credit card numbers are validated using the Luhn algorithm:

  The Luhn Formula:
  - Drop the last digit from the number. The last digit is what we want to check against
  - Reverse the numbers
  - Multiply the digits in odd positions (1, 3, 5, etc.) by 2 and subtract 9 to all any result higher than 9
  - Add all the numbers together
  - The check digit (the last number of the card) is the amount that you would need to add to get a multiple of 10 (Modulo 10)

1. Create a project, Lab06_Q1 in your Lab06 folder. Create a Java class that has the following static methods:

   Notes:
   - For the methods below you should not use Strings or any built-in methods (except for standard input/output methods). You may only use `Math.random()` and `Math.round()/floor()/ceil()` from the Math class.
   - Do not assume the length of the credit card number to validate, different credit cards have different length numbers/even or odd lengths.

   - **`long reverseNumber( long ):`** takes a `long` integer as a parameter and returns the reverse of the number.
   - **`int countDigits( long ):`** takes a `long` integer as a parameter and counts and returns the number of digits in the input number. (Mathematically, not using String or other functionality)
   - **`int sumDigits( long ):`** takes a `long` credit card number as a parameter and sums the digits in the credit card number according to the Luhn formula: starting from the left-most digit, multiply the digits in odd positions (1, 3, 5, etc.) by 2 and subtract 9 from any result higher than 9. Add all the numbers together. You may use the `countDigits()` method in your solution.
   - **`boolean isValidCard( )`**: takes a `long` credit card number as a parameter and returns `true` if the number is valid (according to the Luhn algorithm), `false` if not. Use methods above as appropriate.
   - **`long generateValidCard()`**: generates and returns a *valid* **random** 15 or 16 digit credit card number. Note: you can just generate 15 or 16 *random* digit numbers until a valid number is generated. You may use Math.random() and Math rounding methods here.
   - **`void displayMenu()`**: displays the menu shown in the sample run below.
   - **`main():`**
     - Display the menu below until the user quits. For each choice carries out the requested option.

Sample Run:

```
---------------MENU---------------
1 - Validate Credit Card Number
2 - Generate Valid Credit Card Number
3 - Exit
Enter choice: 1
Enter credit card number to validate: 6011271287933771715
Card number is valid: true

---------------MENU---------------
1 - Validate Credit Card Number
2 - Generate Valid Credit Card Number
3 - Exit
Enter choice: 1
Enter credit card number to validate: 4485043716683962
Card number is valid: true

---------------MENU---------------
1 - Validate Credit Card Number
2 - Generate Valid Credit Card Number
3 - Exit
Enter choice: 1
Enter credit card number to validate: 2221009843129621
Card number is valid: false

---------------MENU---------------
1 - Validate Credit Card Number
2 - Generate Valid Credit Card Number
3 - Exit
Enter choice: 4
Invalid choice - try again

---------------MENU---------------
1 - Validate Credit Card Number
2 - Generate Valid Credit Card Number
3 - Exit
Enter choice: 2
Sample valid credit card: 6090537409995460

---------------MENU---------------
1 - Validate Credit Card Number
2 - Generate Valid Credit Card Number
3 - Exit
Enter choice: 3
Goodbye!
```