Seneca College **Sep 13, 2019**

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

JAC444 Demo & Final Code Due date : Sep 20, 2019

Workshop 1

Notes:

i. Each task should be presented during the lab, demo worth 70% of the workshop marks and code uploading worth the other 30%.

ii. Make sure you have all security and check measures in place, like wrong data types etc., no need to implement Exception as we haven't covered yet. There are other ways to handle bad input data.

iii. Given output structure is just for student to have a glimpse what the output can look, student are free to make the output better in any way.

iv. The final should be submitted by the midnight to avoid late penalties which are 10% each day late.

Other inputs can be given during demo, so make sure you test your program properly.

Task 1: 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

In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. The algorithm is useful to determine whether a card number is entered correctly or whether a credit card is scanned correctly by a scanner. 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. 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.

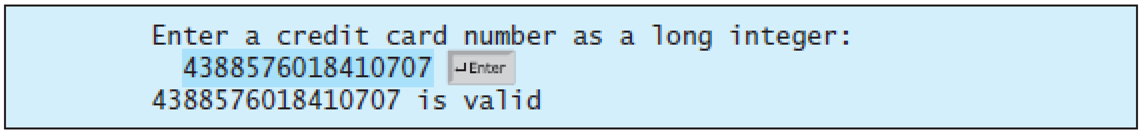
   37 + 38 = 75

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.

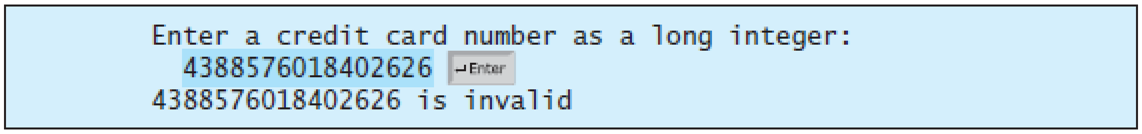
Write a program that prompts the user to enter a credit card number as a long integer (can also use string to solve this). Display whether the number is valid or invalid. Design your program to create and use the following methods:

```
   1. /** Return true if the card number is valid */
      public static boolean isValid(long number)
   2. /** Get the result from Step 2 */
      public static int sumOfDoubleEvenPlace(long number)
   3. /** Return this number if it is a single digit, otherwise,
* return the sum of the two digits */
      public static int getDigit(int number)
   4. /** Return sum of odd-place digits in number */
      public static int sumOfOddPlace(long number)
   5. /** Return true if the digit d is a prefix for number */
      public static boolean prefixMatched(long number, int d)
   6. /** Return the number of digits in d */
      public static int getSize(long d)
   7. /** Return the first k number of digits from number. If the
   * number of digits in number is less than k, return number. */
      public static long getPrefix(long number, int k)
```

Here are sample runs of the program:



```
Enter a credit card number as a long integer:
  4388576018410707 ↵Enter
4388576018410707 is valid
```



```
Enter a credit card number as a long integer:
  4388576018402626 ↵Enter
4388576018402626 is invalid
```

Marking Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention
- Document all the classes properly
- Not have any debug/ useless code and/ or files in the assignment

Deliverables and Important Notes:

- You are supposed to show up AND hand in your solution in person (run the solution and/or answer related Qs) in lab time.
- In case you don't show up OR hand in/run the required task in the lab, you could submit your final solution (described below) on the due date but note that there would be a 70% penalty! Late submissions would result in additional 10% penalties for each day or part of it.
- In this case, you should zip *only the Java files* to a file named after your Last Name followed by the first 3 digits of your student ID. For example, if your last name is **Savage** and your ID is **354874345** then the file should be named **Savage354.zip.** Finally email your zip file to me at mahboob.ali@senecacollege.ca
- Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the assignments, but the final solution may not be copied from any source.