Northeastern University
CSYE 6200 – Concepts of Object-Oriented Design
Spring 2023, Prof. Jones Yu

Lab 3 - Assignment 1

Due: 11:59 pm, Saturday, February 4

In this assignment, you need to write programs to solve the following problems. The source project of the problems is shared on the Canvas. Please import it to the Eclipse to write your codes.

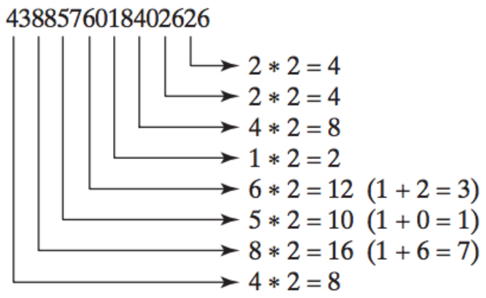
1. Problem 1

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. Display whether the number is valid or invalid. Design your program to use the following methods:

```
/** Return true if the card number is valid */
public static boolean isValid(long number)

/** Get the result from Step 2 */
public static int sumOfDoubleEvenPlace(long number)

/** Return this number if it is a single digit, otherwise, return the sum of the
two digits */
public static int getDigit(int number)

/** Return sum of odd-place digits in number */
public static int sumOfOddPlace(long number)

/** Return true if the digit d is a prefix for number */
public static boolean prefixMatched(long number, int d)

/** Return the number of digits in d */
public static int getSize(long d)

/** 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)
```

Expected results:

```
Enter a credit card number as a long integer: 5117275325077359
5117275325077359 is valid
```

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

Note 1: You may also implement this program by reading the input as a string and processing the string to validate the credit card.

Note 2: You need to use all designed methods in the program. Without using all designed methods will result in losing some points.

2. Problem 2

Write the following method to test whether the array has four consecutive numbers with the same value.

```
public static boolean isConsecutiveFour(int[] values)
```

Write a test program that prompts the user to enter a series of integers and displays if the series contains four consecutive numbers with the same value. Your program should first prompt the user to enter the input size - i.e., the number of values in the series.

Expected results:

```
Enter the number of values: 7
Enter the number: 3 3 5 5 5 5 4
The list has consecutive fours
```

```
Enter the number of values: 9
Enter the number: 3 4 5 5 6 5 5 4 5
The list has no consecutive fours
```

3. Submission Requirement:

This is an individual assignment, and each student needs to submit his/her solution to the Canvas. The submission needs to be a .zip file containing following data:

3.1 A document (either .doc or .pdf format) that describes:

- Problem description:
  - A short description of the issue you are solving in your own words, not simply copy and paste to your report.
  - We would recommend you have a deep thinking in the problem. For example, why does this problem matter? How do I transfer my learning from this problem to other problems? etc.
- Analysis:
  - What design/solution/algorithm do you use to solve the problem?
  - What are the difficulties you encounter?
  - Is there any better solution which might not be used in your solution?
  - ...
- Source code:
  - Copy & paste your source code to the report (i.e., all .java files)
  - The codes need to be readable. (Don't make the color too light.)
- Screenshots of sample runs: show that the code has been reasonably tested

3.2 The project with source codes:

- Source project (.zip file): The Eclipse Java project that has all your changes