

# CSGE601021 Foundations of Programming 2

## (Dasar-Dasar Pemrograman 2)

### Programming Assignment 1:

## Validation of Real Credit Card / Debit Card Numbers

- Deadline for submitting the final result of your work to SCeLE: Tuesday 28 Feb 2023 (11:55 PM SCeLE Time). Use github for storing your partial works.
- Zip all files of your Java project (1 folder) and its runnable JAR file, into one file with name: TP01\_<TAcode>\_<StudentName>\_<NPM>.zip
- Arrange a demo session with your Asdos as soon as possible, while everything about this assignment is still fresh in your memory.
- Start working on this assignment immediately. You will need enough time to understand the problem.
- We learn best by not doing plagiarism.

### Evaluation scheme:

60 % correctness

30 % explanation in demo session

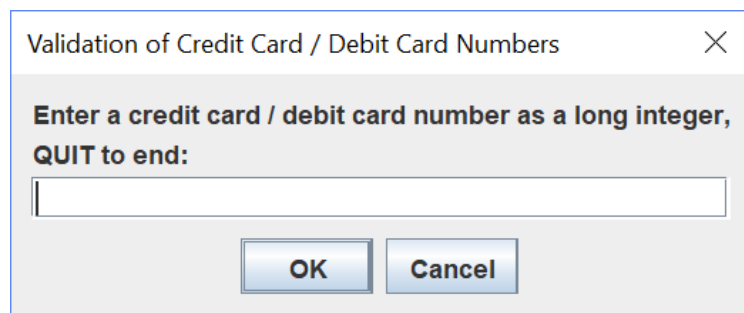
10 % program documentation (comments, neatness)

### Task Description

Build an *Java project* using your favorite IDE and its *runnable JAR file* for the solution of problem 6.31 from the textbook "Introduction to Java Programming and Data Structures", 10th Edition or 11th Edition, by Y. Daniel Liang. A pertinent quote from the book is in the appendix of this document.

Utilize the method `showInputDialog` and method `showMessageDialog` from the class `javax.swing.JOptionPane` like in Lab 02.

### Examples of user interaction:



Validation of Credit Card / Debit Card Numbers ✕

**Enter a credit card / debit card number as a long integer, QUIT to end:**

Validation of Credit Card / Debit Card N... ✕

**The number 4388576018410707 is valid**

Validation of Credit Card / Debit Card Numbers ✕

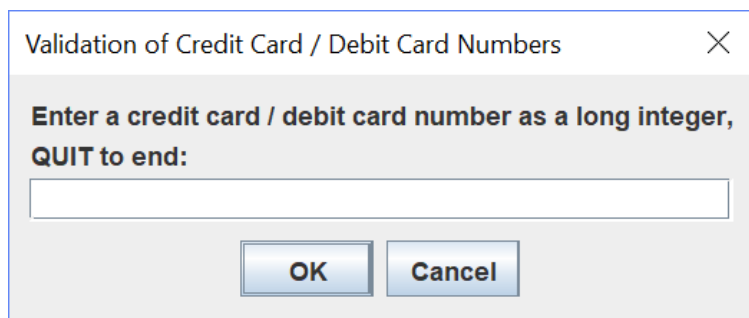
**Enter a credit card / debit card number as a long integer, QUIT to end:**

Validation of Credit Card / Debit Card Numbers ✕

**Enter a credit card / debit card number as a long integer, QUIT to end:**

Validation of Credit Card / Debit Card N... ✕

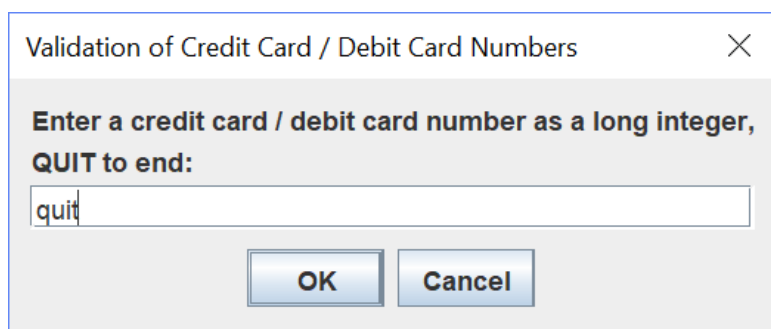
**The number 4388576018402626 is invalid**



Validation of Credit Card / Debit Card Numbers

Enter a credit card / debit card number as a long integer,  
QUIT to end:

OK Cancel



Validation of Credit Card / Debit Card Numbers

Enter a credit card / debit card number as a long integer,  
QUIT to end:

OK Cancel

Try to validate your own credit card / debit card numbers.

**Selamat Mengerjakan!**

Met Ngoding! 😊

L.Y.Stefanus & the Asdos Team

## Appendix

Problem 6.31 from the textbook :

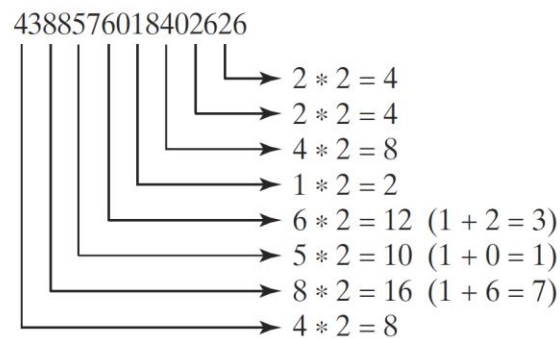
Y. Daniel Liang. "Introduction to Java Programming and Data Structures", 11th Edition, Pearson Education, 2019.

**6.31** (*Financial: credit card number validation*) 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 number 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)
```