BIRZEIT UNIVERSITY

COMP231 – Advanced Programming

Assignment # 1

Objectives:

1. To understand basic problem solving techniques using Java Programming Language.
2. To declare Java primitive data types. In addition, to obtain input and display output.
3. To be able to use arithmetic and use different control structures (Decision making and Loops).

Specification

Submission: through Ritaj.

What to submit: Your **own** **well-structured** and **well-commented** JAVA files (.java) (compressed into a **studentId_labSec#.rar** file, e.g. 1134567_lab3.rar).

Deadline: **28/2/2019** by midnight.

Important

- **The assignment solutions are found in the internet.**
- Academic **honesty**:
  - This is an individual assignment. Individual assignments must be each student's own work.
  - Copying 1 line from a friend or the internet will be considered cheating.
  - Cheating will result in an official university disciplinary review and the University regulations will be strictly enforced.

Tasks

Task 1: Pascal triangle

Write a program that displays a Pascal triangle. The program prompts the user to enter the number of rows and displays the triangle. Here is a sample run:

```
Input number of rows: 8
          1
        1 1
       1 2 1
      1 3 3 1
     1 4 6 4 1
   1 5 10 10 5 1
  1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```

For information on Pascal triangle, see http://www.mathsisfun.com/pascals-triangle.html.

Task 2: Luhn check

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 if a card number is entered correctly, 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 * 2 = 4
   - 2 * 2 = 4
   - 4 * 2 = 8
   - 1 * 2 = 2
   - 6 * 2 = 12 (1 + 2 = 3)
   - 5 * 2 = 10 (1 + 0 = 1)
   - 8 * 2 = 16 (1 + 6 = 7)
   - 4 * 2 = 8
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.
Here are sample runs of the program:

Sample 1:

Enter a credit card number as a long integer: 4246345689049834
4246345689049834 **is invalid**

Sample 2:

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

Good Luck!