# Methods Assignment 1

## Business Overview

You have been asked to help a retail company in Canada to setup their **customer information** and **sales analysis system**.

Your solution should be as robust (defect free) as possible.

Your source code should be: modular, readable, and use coding standards for the language.

Internal documentation should be provided for methods/functions, and classes.

(See the Code Review rubric for details)

An example user interface is below:

Customer and Sales System

1. Enter Customer Information

2. Generate Customer data file

3. Report on total Sales (Not done in this part)

4. Check for fraud in sales data (Not done in this part)

9. Quit

Enter menu option (1-9)

In this project, you will be given the main of the program. Read closely into the program and to see which components can be edited. In this project, there will be **minimal** changes done to the main.

This project will be done in **partners.** This is because you will be taking a “Divide and Conquer” strategy where each member will select some of the prewritten methods in the main. You can also propose to create additional methods in the program to further break down the problem.

Despite some of these methods having dependencies with each other, it’s important for you to communicate with your partner and determine what the proposed inputs and outputs of each method is so that you can each work concurrently.

**The methods will be marked individually. Projects without a work breakdown submissions will NOT be marked.**

**Customer System**

The software system should provide an easy to use interface for employees to enter customer information including the following information for each customer:

* First name (String)
* Last name (String)
* City (String)
* Postal Code (3 or more characters – validation is required)
* Credit Card number (9 or more characters – validation is required)

Some important requirements of the system include:

1. User should be able to enter customer information (using the keyboard or GUI interface)
   1. Postal Codes must be validated
      1. Validation rules
         1. Must be at least 3 characters in length
         2. The first 3 characters must match the postal codes loaded from the file “**postal\_codes.csv**” a | delimiter is used for each field
   2. Credit card numbers must be validated
      1. Validation rules:
         1. Must be at least 9 digits in length
         2. The digits must pass the Luhn algorithm.
   3. The system should automatically assign a unique customer number to each customer starting with an id value of 1
2. User should be able to generate a (Comma Separated Values) CSV output file for all of the customer information (including their assigned id value). User should be able to provide the output file name and location.

## Lunh Algorithm description

The Luhn algorithm test is used as a credit card digit test for many companies. It helps determine if a user accidently enters an incorrect credit card number.

To valid an existing credit card, perform the following test:

1. Reverse the order of the digits in the number.
2. Perform a partial sum of the odd digits (digits 1, 3, etc) – sum1
3. Take the second, fourth ... and every other even digits in the reversed digits
   1. Multiply each digit by two and sum the digits, if the answer is greater than 9 then add the 2 digits to form partial sums for the even digits
   2. Sum the partial sums of the even digits to form sum2
4. If sum1 + sum2 ends in zero then the original number is valid, otherwise it is invalid.

Examples:

|  |  |
| --- | --- |
| **Not Valid** | **Valid** |
| 123456789  Reverse the digits  987654321  Sum the odd digits:  9 + 7 + 5 + 3 + 1 = 25 = sum1  Even digits: 8, 6, 4, 2  Double digits: 16,12, 8, 4  Sum the digits>9: 7, 3, 8, 4  Sum the new digits:7 +3 +8 +4=22=sum2  sum1 + sum2 = 47  **Not valid as the sum does not end with a zero (0).** | 49927398716  Reverse the digits:  61789372994  Sum the odd digits:  6 + 7 + 9 + 7 + 9 + 4 = 42 = sum1  Even digits: 1, 8, 3, 2, 9  Double digits: 2,16, 6, 4, 18  Sum the digits>9: 2, 7, 6, 4, 9  Sum the new digits: 2 +7 +6 +4 +9=28=sum2  sum1 + sum2 = 70  **Valid as the sum ends with zero (0)** |