# CMPP269 Assignment #2 – Employee Salary Program

## Objective

The main purpose of this assignment is to write a program that uses loops, file I/O, and methods to process employee salary data and generate a small report. Additionally, you will gain further practice with several Java concepts that you have already worked with including: strings and string-handling functions; console input and output; and conditional statements.

## Instruction

Your team has been asked to develop a program that receives two data filenames and an employee name (i.e. first & last names) as user input, searches both files for the specified employee's information, and produces a formatted salary schedule for that employee. Separate static methods should be used in your program, as described further in the Solution Design section.

## Input Files

Both input files are plain text files in comma separated value (CSV) format. The first line of each file is a header that can be skipped over by your program.

Here are a few lines extracted from each input file to show the layout.

**Employee.csv:**

```
EmpFirstName,EmpLastName,BirthDate,EmpID,ServiceStartDate
Zoe,Abbott,4/10/1969,E55049,6/25/2020
Amaya,Adams,12/30/1968,E55042,5/29/2020
…
Dolan,Bauer,6/26/1960,E55001,1/5/2018
…
```

**EmpSal2021.csv:**

```
EmpID,MonthlySalary,Fed Tax,AB Tax,CPP,EI
E55001,5600,681.85,365.8,289.3,88.48
E55002,10000,1679.45,812.5,529.07,158
…
```

Information in the two files is related and the common element, Employee ID or EmpID, is what ties the data together. For example, looking first at the Employee.csv sample above, we can find out that the EmpID for Dolan Bauer is E55001. We can then locate Dolan's salary information in the EmpSal2021.csv file from the line with the matching EmpId E55001, which shows:

- 5600 – gross monthly salary
- 681.85 – federal monthly income tax
- 365.8 – Alberta monthly income tax
- 289.3 – monthly CPP payment, up to the maximum of 3,166.45 per year in 2021
- 88.48 – monthly EI payment, up to the maximum of 889.54 per year in 2021

CMPP269 Assignment #2 – Employee Salary Program

## Test Plan

Below are 3 separate sample runs which make up the test plan and demonstrate how your program should obtain user input and present results to the user. User input appears as **<u>bold underline:</u>**

```
Enter employee filename (full path): c:/temp/employee.csv
Enter employee salary filename (full path): c:/temp/empsal2021.csv
Enter employee last name: VALDEZ
Enter employee first name: Britanni

Salary schedule for BRITANNI VALDEZ (E55019)
            Gross                                             Net
Month      Salary         Tax         CPP          EI      Salary
-----------------------------------------------------------------
    1    5,200.00      931.45      267.50       82.16    3,918.89
    2    5,200.00      931.45      267.50       82.16    3,918.89
    3    5,200.00      931.45      267.50       82.16    3,918.89
    4    5,200.00      931.45      267.50       82.16    3,918.89
    5    5,200.00      931.45      267.50       82.16    3,918.89
    6    5,200.00      931.45      267.50       82.16    3,918.89
    7    5,200.00      931.45      267.50       82.16    3,918.89
    8    5,200.00      931.45      267.50       82.16    3,918.89
    9    5,200.00      931.45      267.50       82.16    3,918.89
   10    5,200.00      931.45      267.50       82.16    3,918.89
   11    5,200.00      931.45      267.50       67.94    3,933.11
   12    5,200.00      931.45      223.95        0.00    4,044.60
-----------------------------------------------------------------
Total   62,400.00   11,177.40    3,166.45      889.54   47,166.61
```

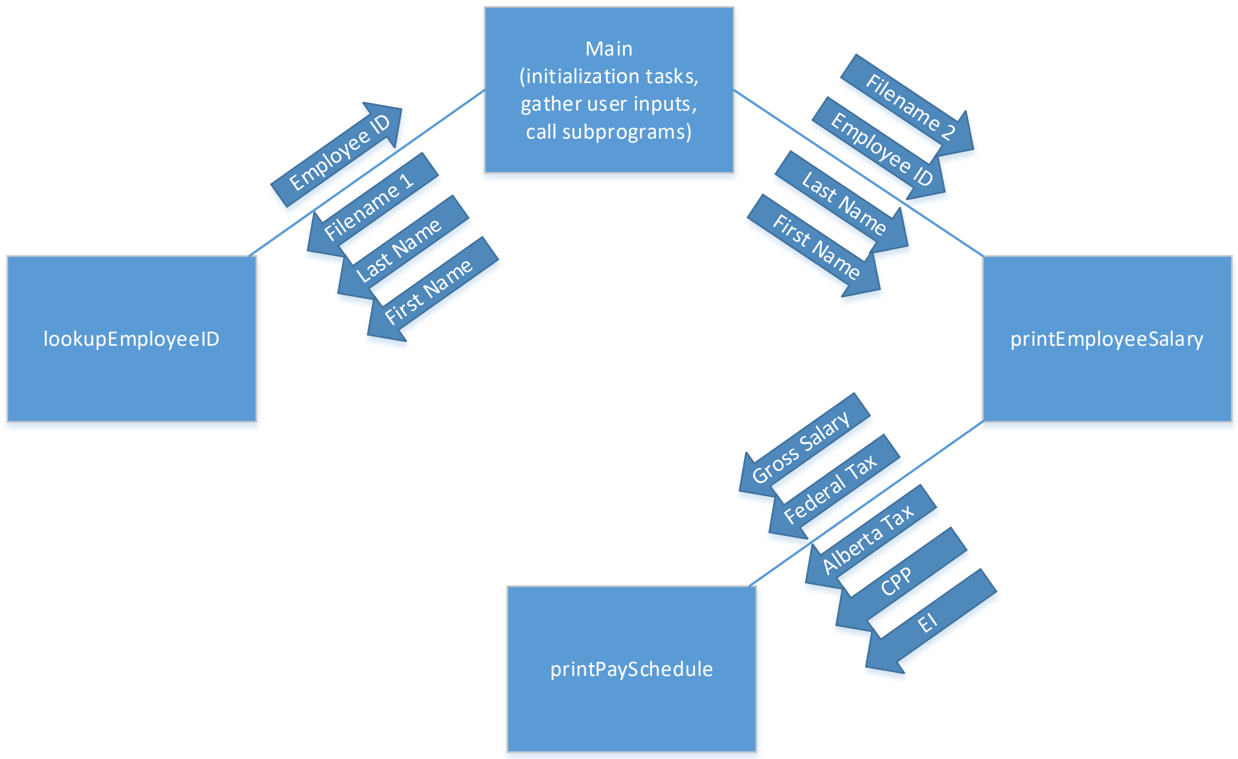
```
Enter employee filename (full path): c:/temp/employee.csv
Enter employee salary filename (full path): c:/temp/empsal2021.csv
Enter employee last name: avery
Enter employee first name: orson

Salary schedule for ORSON AVERY (E55059)
            Gross                                             Net
Month      Salary         Tax         CPP          EI      Salary
-----------------------------------------------------------------
    1    8,100.00    1,809.45      425.52      127.98    5,737.05
    2    8,100.00    1,809.45      425.52      127.98    5,737.05
    3    8,100.00    1,809.45      425.52      127.98    5,737.05
    4    8,100.00    1,809.45      425.52      127.98    5,737.05
    5    8,100.00    1,809.45      425.52      127.98    5,737.05
    6    8,100.00    1,809.45      425.52      127.98    5,737.05
    7    8,100.00    1,809.45      425.52      121.66    5,743.37
    8    8,100.00    1,809.45      187.81        0.00    6,102.74
    9    8,100.00    1,809.45        0.00        0.00    6,290.55
   10    8,100.00    1,809.45        0.00        0.00    6,290.55
   11    8,100.00    1,809.45        0.00        0.00    6,290.55
   12    8,100.00    1,809.45        0.00        0.00    6,290.55
-----------------------------------------------------------------
Total   97,200.00   21,713.40    3,166.45      889.54   71,430.61
```

```
Enter employee filename (full path): c:/temp/employee.csv
Enter employee salary filename (full path): c:/temp/empsal2021.csv
Enter employee last name: Musk
Enter employee first name: Elon
No Employee information found for: Elon Musk
```

## Solution Design

A high level design for the program has been done and your team is expected to adhere to this design, details of which are given below.

Structure chart:



Additional design considerations:

- Use the procedural programming approach (i.e. what has been used in this course so far) rather than an object-oriented approach.
- This program will find and print salary information for a single employee only. In other words the input files will only be read once per execution of the program and you will NOT need to use arrays or other more sophisticated Java data types.
- Keep file I/O code within the appropriate method (i.e. code to process Employee.csv in the lookupEmployeeID method and code to process EmpSal2021.csv in the printEmployeeSalary method).
- Use the Scanner class to read the files as shown in class and in the course material.
- You can read individual values from a CSV file using the token-based Scanner methods (e.g. next, nextDouble) by overriding the default delimiter. Just call the useDelimiter method once on your file Scanner to set this up. For example:

```java
Scanner inData = new Scanner(new File("c:/temp/mydata.csv"));
// Set token delimiter to either a comma or a newline
inData.useDelimiter(",|\r\n");
```

## Group Development

Since this assignment will involve writing multiple methods, it is recommended that your group initially work out what the parameters and return values need to be, using the information provided in the solution design section. A great start would be to define the headers for all of the methods. Then you could divide the programming tasks among your team members and even make use of the "stub" development technique. Once the methods have been written the modules can be put together and then the full program tested and refined.

## Submission and Grading

Please follow the specific submission requirements from your instructor. For a group assignment this would generally mean that one solution, consisting of your group's final Java source code and accompanying text file with output from the test runs, be submitted to Brightspace.

**Submissions will be evaluated according to the following standard:**

1. Working Java code (60%)
    - Follows the high level design provided
2. Style (20%)
    - Indentation – consistent
    - Readability – good variable names
    - Documentation
        - Comments at the top which include: Name, date, and program description including details on inputs, processing and outputs (4-5 sentences minimum).
        - Block comments indicating major code sections and what they do. Including a comment block just before each method is good practice!
3. A sample run that matches the provided test plan (20%)
    - Output formatted according to specification (test plan)