PDP, Fall 2017 Assignment 3 - Airline Company Email Automation

Submission is due October 9th @ 6:00 p.m.

Assignment Goals

- Master I/O – reading and writing data
- Work with regular expressions
- Practice **one class - one responsibility**
- Practice minimizing coupling between classes

Background:

OnTime airlines is a company that values its customers. Every time when there is a delay in departure or arrival a company sends apologies to the passengers of that flight. **You goal is to to automate this process.**

The airline company has all the information of its members in a CSV file. **CSV file** is a plain text file that contains data, such that each piece of data is **enclosed in double quotes and separated by a comma**. The first line of the file contains the headers for each column. For example:

```
"first_name" , "last_name" , "phone"
"James"      , "Butt"      , "504-845-1427"
"Josephine"  , "R, Darakjy", "810-374-9840"
"Art"        , "Venere"    , "856-264-4130"
```

The listing above has 3 columns named `first_name`, `last_name`, and `phone`. The first line contains the headers, while all other lines have the information for a specific passenger. Note that comma serves NOT only as a delimiter, there may be pieces of information with comma, for example `"R, Darakjy"` is one piece of information and not two.

Your task:

Attached to this assignment is a sample of Flight CSV file, which contains first and last names, address, city, country, state, zip, phone, email address and rewards membership. Given such CSV file and a template e-mail (see example below), your task is to **create a program that runs on the command line and generates files that will contain the email messages to all the passengers found in the supplied CSV files.** Templates are stored as text files with **special placeholders** in the text. Placeholders are placed between [[and]]. The placeholders may refer to:

1. CSV file's header names
2. Parameters supplied as an input (see explanation below)
3. Some general information that can be inferred from a system (for example, date of the e-mail).

Please see an example template of an e-mail message on the next page.

```
[[Date]]


To: [[email]]
Subject: Please accept our apologies for the [[event]] of your flight
Dear [[first_name]] [[last_name]],

We are very sorry for the [[event]] of your flight from [[departure-city]] to
[[destination-city]]. As a valued [[rewards]] member of our club your time is
important to us and we will strive to improve our service in the future, and
make it on time!

Sincerely,
OnTime airline customer service
```

Given the above email template, we need to substitute information in placeholders to create an e-mail:

- Some of the information is part of lines in the CSV file. For example:

```
"first_name"    , "last_name"   , "email"              , "rewards"
"Art"           , "Venere"      , "a.venere@gmail.com" , "gold"
```

- Some of the information can be inferred from a system (http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html)
- The information on departure and destination cities is a part of csv file name, which name should have the following format:
  `Flight<`**`id`**`>From<`**`departure-city`**`>To<`**`destination-city`**`>.csv`
- Some of the information will be provided as a parameter/ input to the program, as described below:

Your program needs to accept certain arguments at the command line.

`--email-template <file>`	accepts a filename that holds the email template
`--output-dir <path>`	accepts the name of a folder, all output is placed in this folder
`--csv-file <path>`	accepts the name of the csv file to process
`--event <details>`	specifies if the delay refers to departure/arrival

Options take arguments denoted with < …>, for example `--email-template` takes one argument and it is the name of a file, `--output-dir` takes one argument and it is the name of a folder, `--event` takes

one string argument `departure/ arrival, --csv-file` takes on argument and it is the name of a file according to the following format: `Flight<#>From<departure-city>To<destination-city>.csv`

All the command line options are required with corresponding parameters. When an illegal input is provided by user, the program should exit with a helpful error message and a short explanation of how to use the program along with examples.

For example, upon the following input:

```
--email-template --output-dir --csv-file Flight363FromSeattleToBoston.csv --event arrival
```

Your program should generate appropriate error:

```
Error: --email-template and --output-dir were given without providing appropriate arguments:

Usage:

--email-template <file>       accepts a filename that holds the email template.

--output-dir <path>           accepts the name of a folder, all output is placed
                              in this folder

--csv-file <path>             accepts the name of the csv file to process, in
                              a following format
                              Flight<id>From<departure-city>To<destination-city>.csv

--event <details>             specifies if the delay refers to
departure/arrival

For example:

--email-template email-template.txt --output-dir emails --csv-file Flight363FromSeattleToBoston.csv --event arrival
```

Or upon:

```
--email-template email-template.txt --output-dir emails --csv-file Flight363.csv --event arrival
```

Your program should generate appropriate error:

```
Error: --csv-file argument does not contain departure-city and destination-city

Usage:

--email-template <file>       accepts a filename that holds the email template.

--output-dir <path>           accepts the name of a folder, all output is placed
                              in this folder

--csv-file <path>             accepts the name of the csv file to process, in
                              a following format
                              Flight<id>From<departure-city>To<destination-city>.csv


--event <details>             specifies if the delay refers to
                              departure/arrival
```

For example:

```
--email-template email-template.txt --output-dir emails --csv-file Flight363FromSeattleToBoston.csv --event arrival
```

NOTE the order that the command line options are given does not matter. For example, beyond the above example, the following examples are also valid:

```
--csv-file Flight363FromSeattleToBoston.csv --email-template email-template.txt --output-dir emails --event arrival
```

```
--event arrival --output-dir emails --csv-file Flight363FromSeattleToBoston.csv --email-template email-template.txt
```

Summary of the assignment:

- Design and implement the email generator program for the airline company.
- **You are NOT allowed to use `String` methods for searching and replacing placeholders – use regular expressions.**
- Use the supplied example files to help you develop and test your code. Your code should be flexible enough to work with any CSV file with appropriate naming.
- You program should run for ANY template, with placeholders specified in [[ ]]. If you cannot determine placeholder information, come up with appropriate exception. You can come up with your own templates to test that (beyond the provided email-template.txt).
- **BONUS: send an actual e-mail to YOURSELF, DIRECTLY from your Java Code.** You will need to create a new CSV file with your data.
- Make sure that your program works correctly regardless of how your operating system represents paths and files.
- Come up with a brief write-up, which summarizes main relations between classes and handling all the errors. Please comment, if your program allows for sending actual e-mails.

GOOD LUCK!