CS5004 SPRING 2022

Assignment 8

Refer to Canvas for assignment due dates for your section.

Objectives:

- Continue to familiarize yourself with data collections in Java.
- Command line input argument parsing.
- Java I/O.
- Continue to meet the objectives of previous assignments, where applicable.

General Requirements

Create a new Gradle project for this assignment in your ***group*** GitHub repo.

For this assignment, please continue to use packages as in the past. There is only one problem so you may have only one package but it can be helpful to create additional packages to keep your code organized. The requirements for repository contents are the same as in previous assignments.

GitHub and Branches

Each individual group member should create their own branch while working on this assignment. Only merge to the master branch when you're confident that your code is working well. You may submit pull requests and merge to master as often as you like. Guidelines on working with branches are available from the assignment page in Canvas.

To submit your work, push it to GitHub and create a release on your master branch. Only one person needs to create the release.

The problem: nonprofit communication automation

In Assignment 5, you wrote part of a system that would help a nonprofit track donations. For this assignment, your team will develop software to help the nonprofit automate communication with their supporters. A supporter is defined as someone who has signed up to receive emails or letters from the nonprofit.

The nonprofit stores information about their supporters in CSV file. The CSV file is a plain text file, containing data organized into columns separated by a comma. The data in each column is enclosed in double quotes. The first line of the file contains the headers for each column.

For example, the following listing has 4 columns named `first_name`, `last_name`, `company_name`, and `email`. The second line has information for supporter James Reign.

```
"first_name","last_name","company_name","email"
"James, "Reign","Benton","james.reign@gmail.com"
"Josephine","R, Darakjy","Canay","josie55@hotmail.com"
"Art","Venere","Chemel","art2smart@gmail.com"
```

While the information is enclosed in double quotes, and separated by comma, it is possible that column entries may contain a comma. For example, on row 3 in the listing above, "R,Darakjy" is one valid piece of information, not two.

A sample file, containing some of the nonprofit's supporters' information, is available on Cavas. The CSV file contains first and last name, company name, address, city, county, state, zip, phone 1, phone 2, email address, and web page URL.

The company uses templates for generating communication to be sent to all supporters. The templates are stored as text files with special placeholders in the text that refer to the CSV file's header names. Placeholders are CSV column headers between `[[` and `]]` e.g. `[[first_name]]`.

The nonprofit would like you to create a program that they can run on the command line. The program should take a CSV file and a template (or two) as input, and generate files that will contain the email messages and letters to send to their members. When the program is run, it should output a new text file per row in the CSV file, with all placeholders replaced with the appropriate value for that row. See the "Example input and output" section of this specification for examples.

Your program should accept the following command line arguments[1] in <u>any</u> order:

```
--email                              Generate email messages. If this
                                     option is provided, then --email-
                                     template must also be provided.

--email-template <path/to/file>      A filename for the email
                                     template.

--letter                             Generate letters. If this option
                                     is provided, then --letter-
                                     template must also be provided.

--letter-template <path/to/file>     A filename for the letter
```

[1] You are expected to write your own command line parser, rather than using an external package such as Apache Commons CLI.

	`template.`
`--output-dir <path/to/folder>`	`The folder to store all generated files. This option is required.`
`--csv-file <path/to/file>`	`The CSV file to process. This option is required.`

Note that some options take arguments. For example, `--email-template` takes a file path and `--output-dir` takes a folder. Where an argument is required, it must immediately follow the option. Other options take no arguments and indicate an action e.g. `--email` indicates that email messages should be generated.

Also note that some options require other options to also be present. For example, if the program is run with the `--email` option, then the `--email-template` option (with it's required argument) must also be provided. Calling your program with invalid combinations of arguments should generate an error. For example, the following command is illegal because it contains `--email` but `--email-template` is not provided:

```
--email --letter-template letter-template.txt --output-dir letters/
```

A user can request both emails and letters as long as all the necessary inputs are provided.

When a user provides an illegal combination of inputs, the program should exit with a helpful error message, and a short explanation of how to use the program along with examples. See the "Example input and output" section for an example error message.

For this assignment, you are provided with one example CSV file, and two examples of templates (available on Canvas). The nonprofit, however, may in future like to write more templates, and your program should accommodate those new templates. **Therefore, your code should work for any CSV file and any template that uses the CSV file's header values as placeholders.** Please also make sure that your program works correctly regardless of how your operating system represents paths and files.

Example input and output

Email template

Here is the example template for an email.

```
To:[[email]]
Subject: Spring sale!
Dear [[first_name]] [[last_name]],
```

```
Everything in our store will be 20% off between now and the end of
April! Stock up on our logo mugs, T shirts, and water bottles to show
your support and help raise awareness. Our magnets, plushies, and
picture books, also make great gifts for the children in your life.

Remember, all proceeds go to support our work and, if we can reach
our goal of $10,000 in sales by the end of April, an anonymous donor
has pledged to match every $1 you spend. Want to help out but don't
want to buy stuff? Visit our website to make a donation.

Sincerely,

Non-Profit Director
```

So, given the above email template and the following line from the CSV file:

```
"first_name","last_name","company_name","address","city","county","st
ate","zip","phone1","phone2","email","web"
"Art","Venere","Chemel, James L Cpa","8 W Cerritos Ave
#54","Bridgeport","Gloucester","NJ,"08014","856-636-8749","856-264-
4130","art@venere.org","http://www.chemeljamescpa.com"
```

...the email that is generated looks like:

```
To:art@venere.org
Subject: Spring sale!
Dear Art Venere,
Everything in our store will be 20% off between now and the end of
April! Stock up on our logo mugs, T shirts, and water bottles to show
your support and help raise awareness. Our magnets, plushies, and
picture books, also make great gifts for the children in your life.

Remember, all proceeds go to support our work and, if we can reach
our goal of $10,000 in sales by the end of April, an anonymous donor
has pledged to match every $1 you spend. Want to help out but don't
want to buy stuff? Visit our website to make a donation.

Sincerely,

Non-Profit Director
```

Letter template

Here is the example template for a letter.

```
[[first_name]] [[last_name]]
[[address]]
[[city]], [[state]], [[zip]]
Dear [[first_name]] [[last_name]],
Everything in our store will be 20% off between now and the end of
April! Stock up on our logo mugs, T shirts, and water bottles to show
your support and help raise awareness. Our magnets, plushies, and
picture books, also make great gifts for the children in your life.

Remember, all proceeds go to support our work and, if we can reach
our goal of $10,000 in sales by the end of April, an anonymous donor
has pledged to match every $1 you spend. Want to help out but don't
want to buy stuff? Visit our website to make a donation.

Sincerely,

Non-Profit Director
```

So, given the above letter template and the following line from the CSV file:

```
"first_name","last_name","company_name","address","city","county","st
ate","zip","phone1","phone2","email","web"
"Art","Venere","Chemel,James L Cpa","8 W Cerritos Ave
#54","Bridgeport","Gloucester","NJ","08014","856-636-8749","856-264-
4130","art@venere.org","http://www.chemeljamescpa.com"
```

...the letter that is generated looks like:

```
Art Venere
8 W Cerritos Ave #54
Bridgeport, NJ, 08014
Dear Art Venere,
Everything in our store will be 20% off between now and the end of
April! Stock up on our logo mugs, T shirts, and water bottles to show
your support and help raise awareness. Our magnets, plushies, and
picture books, also make great gifts for the children in your life.

Remember, all proceeds go to support our work and, if we can reach
our goal of $10,000 in sales by the end of April, an anonymous donor
```

```
has pledged to match every $1 you spend. Want to help out but don’t
want to buy stuff? Visit our website to make a donation.

Sincerely,

Non-Profit Director
```

Example error message

If the program is run with the following arguments:
`--email --letter-template letter-template.txt --output-dir letters/`

Then the error message might look something like this:

```
Error: --email provided but no --email-template was given.

Usage:
--email Generate email messages. If this option is provided, then --
email-template must also be provided.
--email-template <path/to/file> A filename for the email template.
--letter Generate letters. If this option is provided, then --letter-
template must also be provided.
--letter-template <path/to/file> A filename for the letter template.
--output-dir <path/to/folder> The folder to store all generated
files. This option is required.
--csv-file <path/to/folder> The CSV file to process. This option is
required.

Examples:
--email --email-template email-template.txt --output-dir emails --
csv-file customer.csv

--letter --letter-template letter-template.txt --output-dir letters -
-csv-file customer.csv
```