CS5010, Fall 2019

Assignment 3

Therapon Skoteiniotis[1] and Tamara Bonaci

t.bonaci@northeastern.edu

- **This assignment is due by 6pm on Monday, October 7. Please push your individual solutions to your designated repo within the CS5010 GitHub organization, tagged as A3-final.**

Assignment objectives:

After completing this assignment, you will have gained:

- Increased familiarity with data collections in Java
- Command line input argument parsing
- Java I/O
- Regular expressions

1. Resources

You might find the following online resource useful while tackling this assignment:

- Java 8 Online Documentation (https://docs.oracle.com/javase/8/docs/api/)
- Java Tutorial: Collections (https://docs.oracle.com/javase/tutorial/collections/index.html)
- Java Tutorial: Basic I/O (https://docs.oracle.com/javase/tutorial/essential/io/index.html)
- Java Tutorial: The Platform Environment (https://docs.oracle.com/javase/tutorial/essential/environment/index.html)
- Java Tutorial: Regular Expressions (https://docs.oracle.com/javase/tutorial/essential/regex/index.html)

2. Submission Requirements

2.1. Gradle Software Management Tool

In this assignment, we introduce the use of Gradle software management tool. Gradle is a tool that allows us to manage our source code in order to:

- Compile that code
- Compile our test code
- Run our test code

- Analyze our source code
- Generate reports about our source code
- Package our software
- Deploy/share our software

Starting with this assignment (and in all lab and homework assignments going forward) you are expected to use Gradle, with the Gradle configuration introduced in this assignment.

2.2.1. Gradle and IntelliJ

IntelliJ has a special project type for Gradle projects.

Please follow these steps to setup the project layout for Gradle projects.

1. Start IntelliJ and create a new project.
2. In the selection window, in the left-hand pane, **instead of selecting Java, select Gradle**.
3. Once you select Gradle in the left-hand pane, the right-hand pane populates with additional libraries and frameworks. Make sure to select "**Java**", and click "`Next`".
4. You will see a dialog box, asking you to enter "**GroupId**" and "**ArtifactId**".
    1. **ArtifactID**: enter the project name (such as "Assignment3")
    2. Ignore the **GroupId** box.
5. On the next screen, make sure that the selected "Gradle JVM" is Java 1.8, and accept the defaults for everything else.
6. Accept the defaults on the next screen too, and click "`Finish`".

At this point, you should have a Gradle project created, but you are not quite done yet.

The last step is to configure your **`build.gradle`** file so that it does everything we want it to do in this course. The easiest way to do this is to open the "**`build.gradle`**" file in your project directory, and copy/paste the file below into that file (the text with the dark background) (this **`build.gradle`** file is also available on Piazza and the course website for your convenience).

```
plugins {
    id 'java'
    id 'pmd' // PMD: source code analyzer to find common programming flaws
    id 'jacoco' // Code coverage
}

defaultTasks 'clean', 'build', 'javadoc', 'check', 'test'

apply plugin: 'java'

group = 'edu.neu.khoury.pdp'
version = '1.0-SNAPSHOT'
description = 'PDP Fall 2019 Seattle'
sourceCompatibility = '8'
sourceCompatibility = 1.8

repositories {
    jcenter()
    mavenLocal()
}
```

```
dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'
    testImplementation 'junit:junit:4.12'

}

jacoco {
    toolVersion = "0.8.4"
    reportsDir = file("$buildDir/customJacocoReportDir")
}

jacocoTestReport {
    reports {
        xml.enabled false
        csv.enabled false
        html.destination file("${buildDir}/jacocoHtml")
    }
}

jacocoTestCoverageVerification {
    violationRules {
        rule {
            limit {
                minimum = 0.3
            }
        }

        rule {
            enabled = false
            element = 'CLASS'
            includes = ['org.gradle.*']

            limit {
                counter = 'LINE'
                value = 'TOTALCOUNT'
                maximum = 0.3
            }
        }
    }
}



test {
    useJUnit()

    maxHeapSize = '1G'
}

tasks.withType(Pmd){
    reports{
        xml.enabled=true
        html.enabled=true
    }
}
pmd {
    ignoreFailures = true
    pmdTest.enabled=true
    ruleSets = [
            "category/java/bestpractices.xml",
            "category/java/errorprone.xml",
            "category/java/codestyle.xml"
    ]
}

tasks.withType(JavaCompile) {
    options.encoding = 'UTF-8'
}

task docs(type: Javadoc) {
    source = sourceSets.main.allJava

}

check.dependsOn jacocoTestCoverageVerification
jacocoTestReport.mustRunAfter test

task doAll{
    dependsOn test
    dependsOn check
    dependsOn javadoc
    dependsOn build

    doLast {
        println 'all done!'
    }
}

jacocoTestReport {
    doLast {
        println "file://$buildDir/jacocoHtml/index.html"
    }
}

javadoc {
    doLast {
        println "file://$buildDir/docs/javadoc/index.html"
    }
}

test {
    finalizedBy jacocoTestReport
}
```

2.2. Code Criteria

- **Naming convention:** Your package name should follow this naming convention assignmentN, where you replace N with the assignment number, e.g., all your code for this assignment must be in a package named assignment3.
- **Gradle built:** Your project should successfully build using the provided **`build.gradle`** file, and it should generate all the default reports.
- **Javadoc generation:** Your Javadoc generation should complete with no errors or warnings.
- **Checkstyle report:** Your Checkstyle report must have no violations.
- **Code coverage report:** Your JaCoCo report must indicate 70% or more code coverage per package for "Branches" **and** "Instructions".

- **Methods hashCode(), equals(), toString():** all of your classes *have to* provide appropriate implementations for methods:

  - `boolean equals(Object o)`
  - `int hashCode()`
  - `String toString()`

(appropriate means that it is sufficient to autogenerate these methods, as long as autogenerated methods suffice for your specific implementation). Please don't forget to autogenerate your methods in an appropriate order - starting from the ancestor classes, towards the concrete classes.

- **Javadoc:** please include a short description of your class/method, as well as tags from `@params` and `@returns` in your Javadoc documentations (code comments). Additionally, if your method throws an exception, please also include a tag `@throws` to indicate that.

- **UML diagrams:** Please include UML diagrams for the final versions of your designs for every problem. In doing so, please note that you do not have to hand-draw your UML diagrams anymore. Auto-generating them from your code will be sufficient.

Problem: Insurance Company Email Automation

You are a member of an IT team for a nationwide insurance company. Unfortunately, your company recently suffered a devastating data breach. Private and sensitive data of millions of your customers may now be compromised…

The whole company is still grappling with the consequences of the breach, but you know the drill – it is time to inform the customers. The members of the board of the company have decided to send an email informing every customer about the data breach. Then, the company will also send the same information using the regular mail. **They are asking you to help them out, and to automate the process that the insurance company will use to communicate with its customers.**

The CSV File

The insurance company holds some information about their members in a CSV file[1]. CSV file is organized as a plain text file, containing information 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.

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

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

Note 1: while the information is enclosed in double quotes, and separated by comma, within the double quotes, there may exist column entries that contain comma. For example, "**R, Darakjy**" is one valid piece of information, and not two.

A sample file, with some of the insurance company's members information, is available on the course website and on Piazza, and titled insurance-company-members.csv. The CSV file contains first and last name, company name, address, city, county, state, zip, phone1 and phone2, email address and web page URL.

Given this CSV file, the insurance company would like you to create a program that they can run on the command line. The program should take this file as input, and generate files that will contain the email messages and letters to send to their members. Email messages and letters will be generated from **templates**.

The templates are stored as text files with special placeholders in the text that refer to the CSV file's header names. Here are two example templates, one for email and one for letters. Placeholders are placed between [[and]].

[1] This partially explains why they suffered the data breach ☺

```
To:[[email]]
Subject: Insurance company – information about recent data breach
Dear [[first_name]] [[last_name]],
As you may have heard or read, last month we learned that criminals
forced their way into our systems, and stole information about our
customers. Late last week, as part of our ongoing investigation, we
learned that the taken information includes names, mailing addresses,
phone numbers or email addresses.
I am writing to make you aware that your name, mailing address, phone
number or email address may have been taken during the intrusion.
I am truly sorry this incident occurred, and I sincerely regret any
inconvenience it may cause you.
Because we value you as a customer, and because your trust is
important to us, our company is offering you one year of free credit
monitoring through Experian’s ProtectMyID product, which includes
identity theft insurance where available. You will receive more
information about this offer via regular mail.
You can find additional information and FAQs about this incident at
our website. If you have further questions, you may call us at 866-
852-8680.
Thank you for your patience and your loyalty.
Sincerely,
Insurance Company CEO
```

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

```
"first_name","last_name","company_name","address","city","county","state","zip","phone1","phone2","email",
“website”
"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.chemeljameslcpa.com"
```

the email message that is generated looks like:

```
To:art@venere.org
Subject:Insurance company – information about recent data breach
Dear Art Venere,
As you may have heard or read, last month we learned that criminals
forced their way into our systems, and stole information about our
customers. Late last week, as part of our ongoing investigation, we
learned that that information includes names, mailing addresses, phone
numbers or email addresses.
I am writing to make you aware that your name, mailing address, phone
number or email address may have been taken during the intrusion.
I am truly sorry this incident occurred, and I sincerely regret any
inconvenience it may cause you.
Because we value you as a customer, and because your trust is
important to us, our company is offering you one year of free credit
monitoring through Experian’s ProtectMyID product, which includes
identity theft insurance where available. You will receive more
information about this offer via regular mail.
```

```
You can find additional information and FAQs about this incident at
our website. If you have further questions, you may call us at 866-
852-8680.
Thank you for your patience and your loyalty.
Sincerely,
Insurance Company CEO
```

Similarly, we also have a template for the company's letter:

```
[[first_name]] [[last_name]]
[[address]], [[city]],
[[county]], [[state]], [[zip]]
([[email]])

Dear [[first_name]] [[last_name]],
As you may have heard or read, last month we learned that criminals
forced their way into our systems, and stole information about our
customers. Late last week, as part of our ongoing investigation, we
learned that the taken information includes names, mailing addresses,
phone numbers or email addresses.
I am writing to make you aware that your name, mailing address, phone
number or email address may have been taken during the intrusion.
I am truly sorry this incident occurred, and I sincerely regret any
inconvenience it may cause you.
Because we value you as a customer, and because your trust is
important to us, our company is offering you one year of free credit
monitoring through Experian's ProtectMyID product, which includes
identity theft insurance where available. Enclosed please find your
additional information about your free credit card monitoring.


You can find additional information and FAQs about this incident at
our website. If you have further questions, you may call us at 866-
852-8680.
Thank you for your patience and your loyalty.

Sincerely,

Insurance Company CEO
```

For the same CSV lines we used before, this template generates the following text file:

```
Art Venere
8 W Cerritos Ave #54, Bridgeport,
Gloucester, NJ, 08014.
(art@venere.org)

Dear Art Venere,
As you may have heard or read, last month we learned that criminals
forced their way into our systems, and stole information about our
customers. Late last week, as part of our ongoing investigation, we
learned that the taken information includes names, mailing addresses,
phone numbers or email addresses.
I am writing to make you aware that your name, mailing address, phone
number or email address may have been taken during the intrusion.
I am truly sorry this incident occurred, and I sincerely regret any
inconvenience it may cause you.
Because we value you as a customer, and because your trust is
important to us, our company is offering you one year of free credit
monitoring through Experian's ProtectMyID product, which includes
identity theft insurance where available. Enclosed please find your
additional information about your free credit card monitoring.


You can find additional information and FAQs about this incident at
our website. If you have further questions, you may call us at 866-
852-8680.
Thank you for your patience and your loyalty.
Sincerely,

Insurance Company CEO
```

Note 2: in this assignment, you are provided with **one** example CSV file, and **two** examples of templates. The insurance company, however, may in the future like to write more templates, and your program should accommodate for new templates. **Therefore, your code should work for any CSV file and any template that uses the CSV file's header values as placeholders.**

Command Line Arguments

Your program should accept certain arguments at the command line.

```
--email                  only generate email messages
--email-template <file>  accept a filename that holds the email
template

--letter                 only generate letters
--letter-template <file> accept a filename that holds the email
template

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

--csv-file <path>        accept the name of the csv file to
process
```

Please notice that some options take arguments, for example `--email-template` takes one argument, and it is the name of a file. Similarly, `--output-dir` takes one argument, and it is the name of a folder.

Other options, however, take no arguments, and indicate an action, i.e., `--email` indicates that we should generate emails on this execution of the program.

The command line option `--output-dir` and `csv-file` are required.

Your program should be able to generate one of the two options (emails or letters) per invocation. If `--email` is given, then `--email-template` must also be provided, if `--letter` is given then `--letter-template` must also be given.

Calling your program and passing any other combination of options should generate an error, for example:

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

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. For example, passing:

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

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

 Usage:
   --email                        only generate email messages
   --email-template <file>  accept a filename that holds the
email template.
   Required if --email is used

        --letter                        only generate letters
        --letter-template <file> accept a filename that holds
the email template.
          Required if --letter is used

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

        --csv-file <path> accept the name of the csv file to
process

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
```

Finally, please note the order of the command line options does not matter, i.e. the following examples are valid:

```
        --email --email-template email-template.txt --output-dir emails --csv-file customer.csv
        --csv-file customer.csv --email-template email-template.txt --output-dir emails --email
        --output-dir emails --email --csv-file customer.csv --email-template email-template.txt
```

Your tasks:

1. Design and implement the email and the letter generator program for the insurance company, as described above.
2. Please write appropriate Javadoc documentation for all of your classes and methods.
3. Please write the corresponding test classes for all of your classes.
4. Please use insurance-company-members.csv, email-template.txt and letter-template.txt to help you develop and test your code.
5. Please also develop your own templates (one for email, and one for letter), and include it with your submission.
6. Please provide a final UML Class Diagram for your design

Note 3: Please also make sure that your program works correctly regardless of how your operating system represents paths and files.