**Lab Assignment 5  
Contact Information Formatter**

|  |
| --- |
| **Due Date** |
| **Wednesday 11/06/19 @ 11:59pm** |

**Objectives**

* String manipulation
* Exception handling
* Pair programming

**Problem Specification**

Develop a Java application to read contact information from several files, format them and print them to the console.

You are ***required*** to implement the program using **exception handling mechanisms** as specified in this document. Your program should include a contact information formatter which is given, as a parameter, an array of 10 input files (see the main method provided in the Testing Phase of this document). The contact information formatter reads the contact information from the files and performs a format check on each piece of data. If the contact information is in the correct format, the formatter prints out the contact information directly. If the contact information is not correct, the contact information formatter will throw an exception and let the corresponding format exception handler handle this issue (there should be format exception handlers for each type of data: full name, phone number, and email address). The format exception handler will change the contact information into the correct format and print it out.

The correct formats for the full name, phone number and email addresses are defined using regular expression as follows:

1. **Full-name**

[A-Z][a-z]+[ ][A-Z][a-z]+ (For example: **Hao Li**)

1. **Phone number**

([0-9][0-9][0-9])-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9] (For example: **(269)-234-2452**)

The phone number should be divided into three groups with each pair separated by a dash (‘-‘). The area code should be put into parentheses.

1. **Email Address**

[a-z]([a-z])\*(.[a-z]+)\*@[a-z]+.(a-z)+ (For example: **hao.81.li@gmail.com**)

All letters in the email address should be lower case.

The following examples demonstrate how the formatter works.

**The original text from the file**

hAo li

(734272-3020

[Hao.81.lI@gMail.com](mailto:Hao.81.lI@gMail.com)

**The output from the contact information formatter.**

Hao Li

(734)-272-3020

hao.81.li@gmail.com

**Assumptions:**

* You can assume that full names in the input file only have two words: first name and last name. You only need to make sure the letters are in the correct cases (i.e. the first letter of each name should be capitalized).
* You can assume the phone number in the input file always contains 10 digits. It could also contain characters other than digits (e.g. parentheses, dashes).
* You can assume that email addresses in the input files may have only one problem: some of the letters may be in upper case. You just need to convert all the upper case letters into lower case.

**Design Requirements**

Your program should have the following classes:

* ContactInformationFormatter
* NameFormatException (extends Exception)
* EmailAddressFormatException (extends Exception)
* FormatExceptionHandler
* PhoneNumberFormatException (extends Exception)

Note that you will need to write your own exception classes (three of them, as specified above).

The **FormatExceptionHandler** class should implement the following interface;

**public** **interface** IFormatExceptionHandler {

**void** handleFileNotFoundException(FileNotFoundException e);

/\*Handle file not found exception\*/

**void** handlePhoneNumberFormatException(PhoneNumberFormatException e);

/\*Handle phone number format exception\*/

**void** handleEmailFormatException(EmailAddressFormatException e);

/\*Handle email format exception\*/

**void** handleNameFormatException(NameFormatException e);

/\*Handle name format exception\*/

}

The **ContactInformationFormatter** should implement the following interface;

**public** **interface** IContactInformationFormatter {

**void** readContactInformation(String[] filePaths);

/\*read contact information from a list of files given the file paths\*/

**void** formatContactInformation(String fileName);

/\*format the contact information in one file\*/

**void** formatEmail(String email) **throws** EmailAddressFormatException;

/\*format one email address\*/

**void** formatPhoneNumber(String phoneNumber) **throws** PhoneNumberFormatException; /\*Format one phone number\*/

**void** formatName(String name) **throws** NameFormatException;

/\*format one name\*/

}

**Pseudocode**

Write pseudocode for the required methods (as specified above) and for any other methods you may want to use in your code.

*Note:* ***Correct pseudocode development*** *will be worth* ***40%*** *of the total LA grade.*

You will also need to **generate Javadoc** for your project. Follow the steps shown in class (a copy of these steps can be found on the Content page in Elearning, or can otherwise be provided to you by your Lab instructor). If you follow the steps accurately, a “**doc**” folder (for Javadoc) should be created in your project.

**Implementation Phase**

Using the pseudocode developed, write the Java code for your assignment. This is a one-week assignment.

**Testing Phase**

Use the Main class below to test your program. It must **not** be modified.

**public** **class** Main {

**public** **static** **void** main(String[] args) {

String[] filePaths = {"testcase1.txt","testcase2.txt", "testcase3.txt", "testcase4.txt", "testcase5.txt", "testcase6.txt", "testcase7.txt", "testcase8.txt", "testcase9.txt", "testcase10.txt"};

ContactInformationFormatter formatter = **new** ContactInformationFormatter();

formatter.readContactInformation(filePaths);

}

}

**Expected output:**

Hao Li

(734)-272-3020

hao.81.li@gmail.com

Hao Li

(734)-292-3051

hao.81.li@gmail.com

Alex John

(737)-890-8703

alex.john@gmail.com

Mary Ma

(289)-098-4591

mary.ma@gmail.com

Jessica Done

(889)-034-9201

jessica.done@gmail.com

Shirley Hu

(260)-094-3492

shirly.hu@gmail.com

Nick Harry

(250)-943-8720

nick.harry@gmail.com

Nichole Sara

(280)-203-0295

nichole.sara@gmail.com

Jack Ma

(260)-892-8024

jack.ma@gmail.com

testcase10.txt (The system cannot find the file specified)

**Additional Requirements**

**Pair Programming**

You should each have a GItLab account, and should also have a partner with whom you will be working on this assignment.

There should be visible evidence of both partners in a team contributing to the work required for this assignment (each person’s commits should reflect in the team’s repository). You will be graded partially on how well you use your GIT repository. The “commits” on GitLab will be reviewed to determine if there was input from both team members. There should be at least two commits from each person on the team. In addition, in the pseudocode document, a **brief** explanation should be included explaining what each team member worked on.

**You must add your lab instructor (with Master access) to your GIT repository to provide access for grading purposes.** For additional help outside of the explanations provided by your lab instructors, you may consult the “Git Guide” document which your lab instructors will make available to you. It includes steps specifying how to perform various tasks on Git.

If you experience any problems with Git, get in touch with your lab instructor at the earliest opportunity. A zip file containing your submission needs to be submitted to Elearning only if you can’t get Git working (you will be penalized for this, so you need to make every effort to work with Git). If your code is in the repository, you will not have to upload it to Elearning. However, you still need to keep to the due date for submission – note that the most recent date/time of modification can be seen in Git.

**Coding Standards**

You must adhere to all conventions in the CS 1120 Java coding standard (available on Elearning for your Lab). This includes the use of white spaces and indentations for readability, and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions also for naming classes, variables, method parameters and methods.

**Assignment Submission**

* The final version of your project should be uploaded to the GitLab repository (a Git push) by the due date / time.
* The pseudocode should be submitted to the appropriate folder on Elearning. Only one pseudocode document is required for each team / pair and the names of the team members should be clearly indicated on the report.

**NOTE**: The eLearning folder for LA submission will remain open beyond the due date but will indicate how many days late an assignment was submitted where applicable. The dropbox will be inaccessible seven days after the due date by which time no more credit can be received for the assignment.

The penalty for late submissions as stated in the course syllabus will be applied in grading any assignment submitted late.