Assignment #1
CSCI 201 Fall 2019
2.5% of course grade

Title
201 Contacts Book

Topics Covered
Java Classes
File I/O
Sorting Algorithms
Basic Java Topics

Introduction
Much of your computer programming experience is likely using C++. In this course, Java is the language we will primarily use. To help transition your knowledge from C++ to Java, you will be creating a program that populates a contact book and allows the user to search and modify it.

This assignment requires that you parse a file containing a set of contacts and their information. To ensure accurate parsing, you will then provide a command line interface to allow a user to query and modify the parsed data.

Data Persistence
Many software projects need to store persistent data, and there are a few ways to do this. Databases, which we will cover later in the class, are primarily used for most web applications, but storing data in files is also very common for many types of applications (i.e. mobile, standalone, web). This assignment will require you to parse a file that contains contact information. The data in the file is going to be stored as a pipe-delimited file, formatted as follows:

```
firstName,lastName,emailAddress,age,nearCampus,note1,…,noteN
```

Assume every email address is unique. There could be multiple `notes`, but there will be at least one. A sample file is provided here and posted on the course web site. I recommend that you create your own test files that are longer, since the file we will use for testing will be different from the sample one provided.

```
Sam,Smith,samsmith@usc.edu,20,true,201 classmate
Tiffany,Smith,tsmith@usc.edu,18,true,104 classmate
Tommy,Trojan,ttrojan@usc.edu,21,false,USC mascot,horse rider
John,Brown,jbrown@usc.edu,17,false,cousin,best friend
```

Assignment

When your program first runs, you will need to prompt the user to enter the name of the file that contains all the contact information. Your program should validate that the file is formatted properly. All the fields are required. There should be at least one `note`, though there could be more than one. You can assume the following data types for each:

```
firstName – String
lastName – String
emailAddress – String
age – int
nearCampus – boolean
notes – String
```

If there is any problem parsing the data in the contact file, your program should print out as descriptive of an error message as possible. The program should then prompt the user to enter another contact file. Here are the errors in file parsing you need to catch:

- File not found.
- Data cannot be converted to the proper type as shown above.
- Too few parameters on one of the lines.

Once a properly-formatted contact file is parsed, show a menu with different options and prompt the user to select an option:

```
1. Contact look up
2. Add contact
3. Delete contact
4. Print to a file
5. Exit
```

The user should enter a number, such as 3. If the user enters something else, the program should show an error message and prompt the user to enter a valid option.

If the user enters 1:

> Your program should prompt the user to enter the last name of the person they are trying to find. If there are multiple contacts with the same last name, print them in alphabetical order according to their first name.

If the user enters 2:

Prompt the user to enter the first name, last name, email address, age, near campus information, and notes about their new contact.

Be sure to do any error checking (see below for specifications).

If the user enters 3:

Prompt the user to enter the email of the contact they want to delete from their contacts book. You can assume that each contact has a unique email address.

If the user enters an email address of a contact that does not exist, be sure to show an error message and prompt the user to enter a valid one.

If the user enters 4:

Your program should prompt the user to enter a filename to which the entire contacts list will print.

The order of the contacts in the file should be alphabetized according to last name. If two contacts have the same last name, alphabetize them by their first name.

If the user enters `EXIT` or 5, the program will terminate.

All data entered should be case-insensitive, except for EXIT, which should be in all caps.

Error Checking

1. Make sure the information entered is of a correct type. If it is not, display an error message and re-prompt the user. (i.e. entering a string in the age field is not allowed.)
2. For emails, the format should be xxx@yyy.com, xxx@yyy.net, or xxx@yyy.edu. If @ and (.com, .net, or .edu) do not appear in the string in that exact order, the email is considered invalid.
3. Both first and last name should only contain alphabetic letters. If they contain numbers or special characters, it is considered invalid.

Sample Execution

Here is a sample execution of the program with the user input bolded (though the input will not be bolded when you run your program). Assume `goodcontacts.txt` contains the data from the sample provided above.

```
What is the name of the contacts file? nocontacts.txt
The file nocontacts.txt could not be found.

What is the name of the contacts file? badcontacts1.txt
This file badcontacts1.txt is not formatted properly.
The parameter "bademail.com" cannot be parsed as an email.

What is the name of the contacts file? badcontacts2.txt
This file badcontacts2.txt is not formatted properly.
There are not enough parameters on line
'Bad,bad@bad.com,15,false,terrible'.

What is the name of the contacts file? goodcontacts.txt

   1) Contact Lookup
   2) Add contact
   3) Delete contact
   4) Print to a file
   5) Exit

What option would you like to select? hello

That is not a valid option.

What option would you like to select? 1

Enter the contact's last name. Holfand

There is no one with the last name Holfand in your contact
book.

   1) Contact Lookup
   2) Add contact
   3) Delete contact
   4) Print to a file
   5) Exit

What option would you like to select? 1
```

Display this error message if the user enters anything other than the numbers 1-5.

Should be case-insensitive

This should be exactly as the user entered (same case).

Enter the contact's last name. **Smith**

Name: Sam Smith
Email: samsmith@usc.edu
Age: 20
Near Campus: Yes
Notes: 201 classmate

If there are multiple notes, separate them with commas.

Name: Tiffany Smith
Email: tsmith@usc.edu
Age: 18
Near Campus: Yes
Notes: 104 classmate

1) Contact Lookup
2) Add contact
3) Delete contact
4) Print to a file
5) Exit

What option would you like to select? **2**

What is the first name of your new contact? **Bob**

What is the last name of your new contact? **Jenkins**

What is the email of your new contact? **Bobjenkins**

That is not a valid email. An email must have this formatting: xxx@yyy.com

What is the email of your new contact? **bobjenkins@bobjenkins.com**

What is the age of your new contact? **25**

Does your new contact live near campus? **Yes**

Add a note about your new contact. **Bob is a nice man.**

Do you want to add another note? **Yes**

Add a note about your new contact. **Bob is a great man.**

Do you want to add another note? **No**

```
Bob Jenkins has been added to your contact list.

   1) Contact Lookup
   2) Add contact
   3) Delete contact
   4) Print to a file
   5) Exit

What option would you like to select? 3

Enter the email of the contact you would like to delete.
doesntexist@gmail.com

doesn'texist@gmail.com does not exist in your contact list.

Enter the email of the contact you would to delete. samsmith@usc.edu

Sam Smith was successfully deleted from your contact list.

   1) Contact Lookup
   2) Add contact
   3) Delete contact
   4) Print to a file
   5) Exit

What option would you like to select? 4

Enter the name of the file that you would like to print your contact
list to. contactlist.txt

Successfully printed all your contacts to contactlist.txt

   1) Contact Lookup
   2) Add contact
   3) Delete contact
   4) Print to a file
   5) Exit

What option would you like to select? EXIT

Thank you for using my contacts program. Goodbye!
```

Grading Criteria

The manner by which you go about implementing the solution is not specifically graded, but the

output must match exactly what you see in the execution above.

File Parsing (0.5%)
0.1% - the filename is read from the user
0.2% - if the file cannot be parsed, an appropriate error message is displayed
0.2% - if the file cannot be found, an appropriate error message is displayed

Output (2.0%)
0.1% - The user is prompted for a menu option properly.
0.1% - The input is validated correctly and if need be the menu is shown again.
Option 1:
0.05% - Reading a contact that does not exist produces an error message.
0.05% - The user is prompted again to enter the name of a contact.
0.4 % - The contacts with the entered last name are displayed.
0.1% - If there are multiple contacts with the same last name, they are alphabetically ordered according to first name.
Option 2:
0.05% - Entering information that is invalid produces an error message and the user is prompted again.
0.05% - Once the new contact is added, a success message is shown.
0.05% - The new contact is actually added to the list and can be seen when looked up.
0.05% - The new contact is actually added to the list and can be seen when the file is printed.
Option 3:
0.05% - Entering information that is invalid produces an error message and the user is prompted again.
0.05% - Once the contact is deleted, a success message is shown.
0.05% - The contact is removed from the list and cannot be seen when looked up.
0.05% - The contact is removed from the list and cannot be seen when the file is printed.
Option 4:
0.05% - A file with the inputted filename is created.
0.5% - The contacts list is printed to the file.
0.1% - The contacts list is sorted alphabetically by last name, then first name.
0.05% - A success message is shown once the contacts are printed to the file.
Option 5:
0.05% - Exiting the program works using EXIT
0.05% - Exiting the program works using 5