RIT | Golisano College of Computing and Information Sciences
School of Information

ISTE-120 - Computational Problem Solving for the Information Domain I
Homework Assignment 13 (HW13) - Character File IO

Download the HW13Downloads.zip file from myCourses.

Objectives:

1. Open a file for writing
2. Construct a record from variables of data
3. Write a record to a file and close the file
4. Open a file for reading
5. Read a record from a file
6. Populate an array with record fields and close the file

Overview:

Construct a phone book data storage mechanism that stores the data in a character-based file. Each record will contain the following information fields:

- Last name
- First name
- Phone

Specific:

Create a Java application that will allow a user to:

1. search an address file for a particular last name and then display the information associated with that last name. To search the file, open the file for reading, read each record and test the record's field value to see if it is a match
    a. prompt the user for a last name
    b. display all the matching records
2. display all records in the file; open it for reading, read each record, display field values
    a. display the `lastName, firstName, phone`
    b. when all records have been displayed, display the record count (should equal the number of records in the file)
3. add a new record to the file. Open the file for writing in append mode using the FileWriter constructor with the appropriate parameters
    a. prompt user to enter data for each field in the record. The last name is required
    b. do not overwrite existing data. The `FileWriter` class provides an overloaded constructor with a boolean parameter to indicate if data should be appended
4. end the program

Implementation Notes:

Carefully review the starter code provided! Developers frequently work on code that has already been written. It is not necessary to write the entire application. A complete `TestContactList` class and a partial `ContactList` class are provided. Do not change the `TestContactList` class. Code development is done in the `ContactList` class. All of the methods needed have already been declared within the `ContactList` class. The code is commented: **read the code!**

File and Record Structure:

The file record structure has the following sequence of fields, one record per line:

```
Last name, First name, Phone
```

Fields Descriptions:

Field Name	Field Data Type	Comments
`Last name`	`String`	Required
`First name`	`String`	Optional
`Phone`	`String`	Optional, any phone format is acceptable

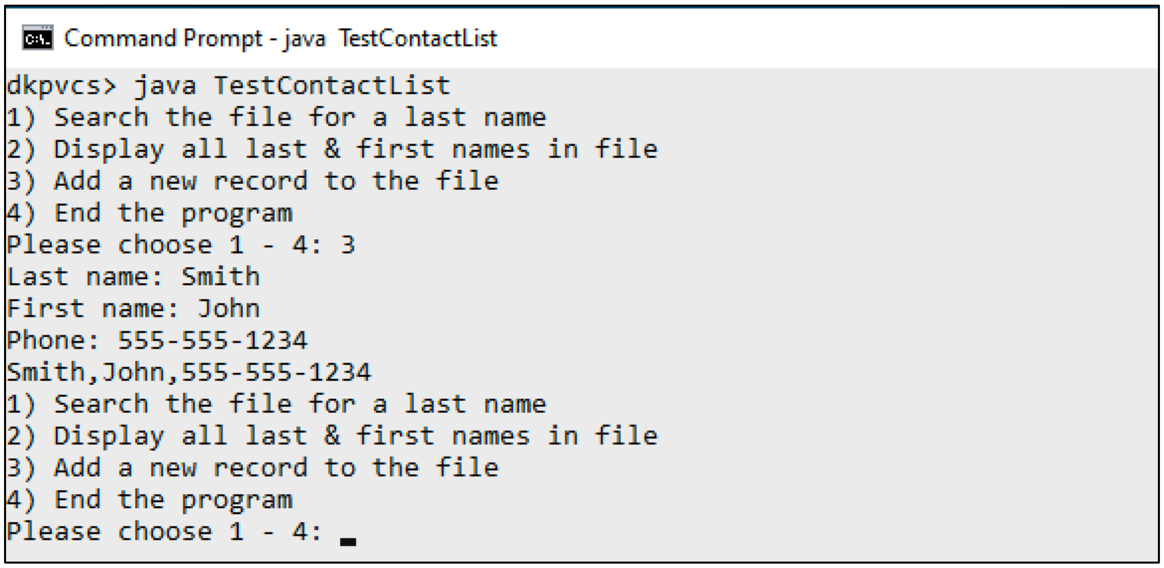
- Records (and fields) are variable length
- Fields are delimited by commas; the last field of each record is not followed by a comma
- Each record is terminated with a "new line" character

Miscellaneous program requirements:

1) The name of the address file must be MyAddressBook.txt
2) The file must be in the same directory as the class that opens the file
3) Do not allow the program to "crash" because of any unhandled exceptions (no "stack traces"). Custom exceptions messages are encouraged. End the program if the exception is unrecoverable
4) Do not allow any completely empty records to be written to the file (the comma delimiters do not count as data). Minimally, the lastName field should have data
5) Javadocs style formatting for all classes, methods, and public attributes is required

Use Cases:

Case 1: Start when no file exists or no data exists within the file. Choice 3 - Adding a record to the file

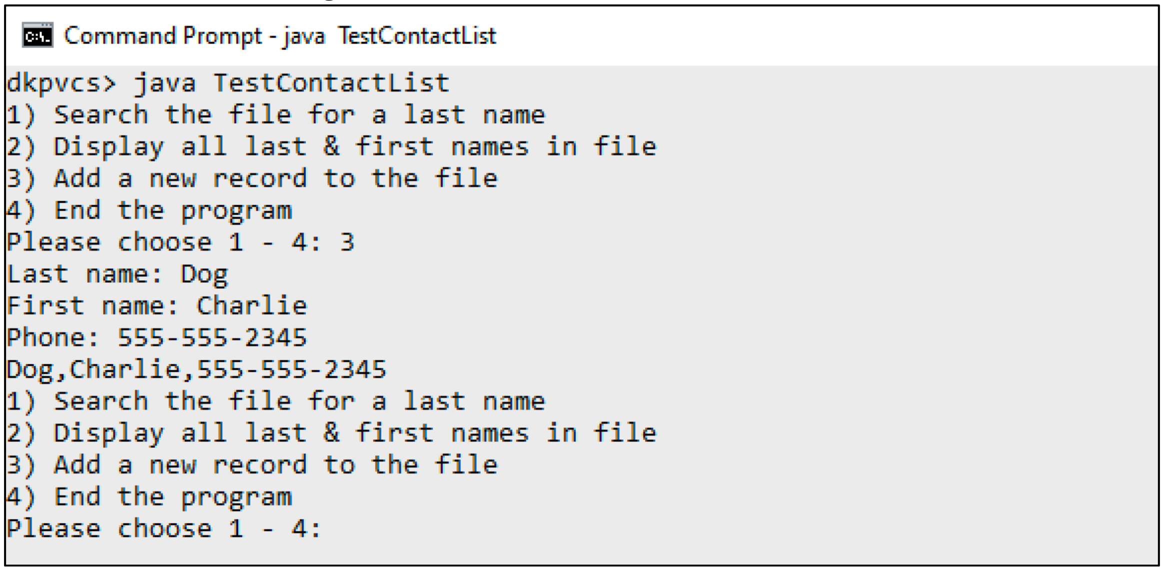


Command Prompt - java TestContactList

```
dkpvcs> java TestContactList
1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4: 3
Last name: Smith
First name: John
Phone: 555-555-1234
Smith,John,555-555-1234
1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4: _
```

The file now has one record:
Smith,John,555-555-1234

Case 2: Choice 3 - Adding another record to the file

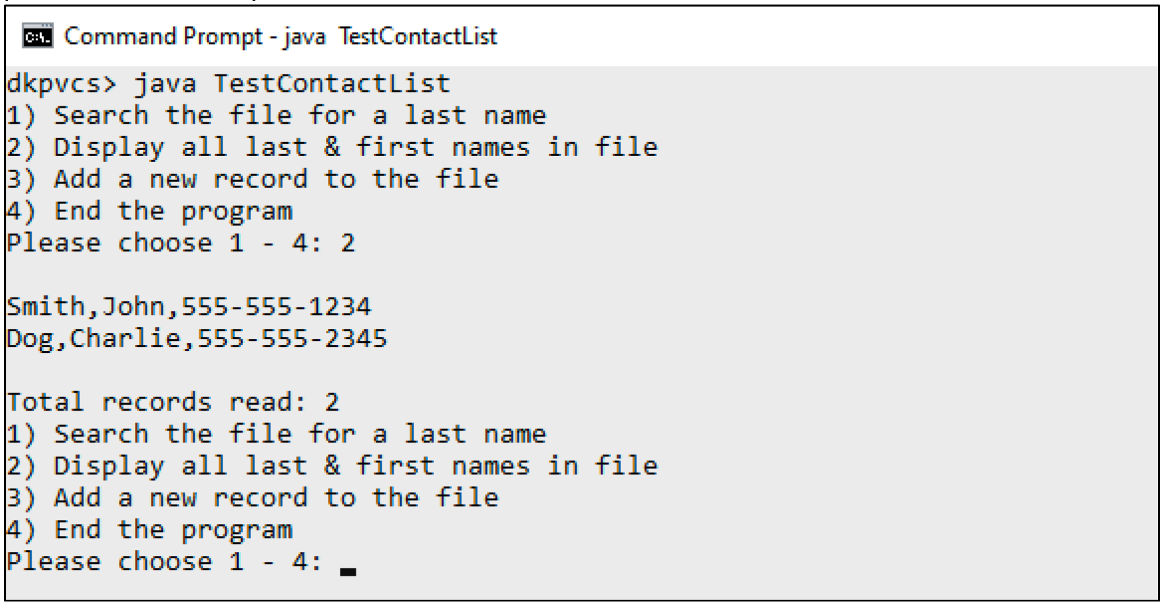


Command Prompt - java TestContactList

```
dkpvcs> java TestContactList
1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4: 3
Last name: Dog
First name: Charlie
Phone: 555-555-2345
Dog,Charlie,555-555-2345
1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4:
```

The file now has two records:
Smith,John,555-555-1234
Dog,Charlie,555-555-2345

Case 3: Choice 2 - Display all records in the file with the following format: lastName, firstName, phone - followed by the number of records read



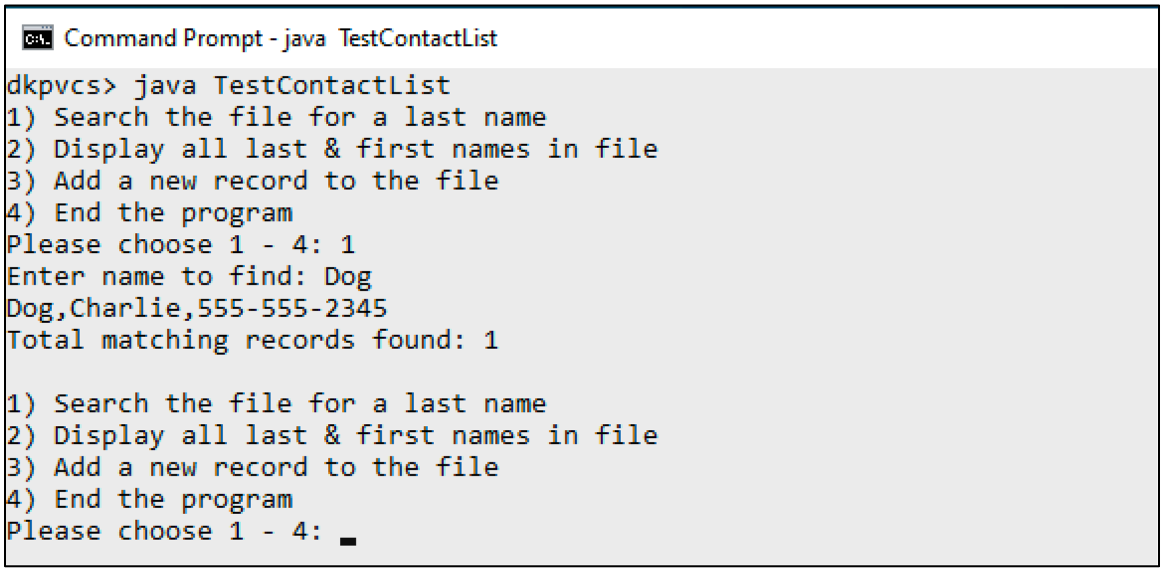
Command Prompt - java TestContactList

```
dkpvcs> java TestContactList
1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4: 2

Smith,John,555-555-1234
Dog,Charlie,555-555-2345

Total records read: 2
1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4: _
```

Case 4: Choice 1 - Search for a last name

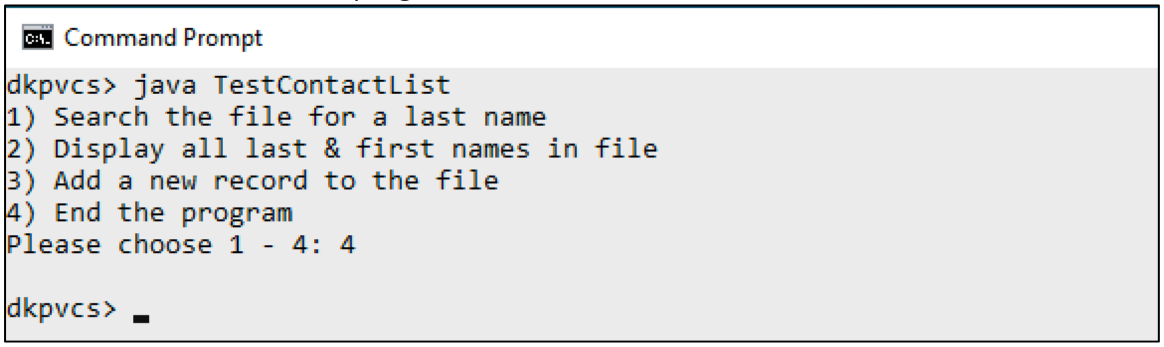


Command Prompt - java TestContactList

```
dkpvcs> java TestContactList
1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4: 1
Enter name to find: Dog
Dog,Charlie,555-555-2345
Total matching records found: 1

1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4: _
```

Case 5: Choice 4 - End the program


Command Prompt

```
dkpvcs> java TestContactList
1) Search the file for a last name
2) Display all last & first names in file
3) Add a new record to the file
4) End the program
Please choose 1 - 4: 4

dkpvcs> _
```

Programming "considerations":

1) Java character I/O classes do not have a mechanism for reading one string field at a time from a file. You will need to use the `split()` method of the String class. The split method is overloaded. One of the split methods will take a String that indicates the character to use as a field delimiter. In our case, the delimiter is the comma character. The split method to use is: `split(",")`
2) The format used to write data to a file determines the format used to read data from a file

RIT | Golisano College of Computing and Information Sciences
School of Information

Grade Sheet Name: ______________________

No points if program does not compile.

Criteria	Max Pts	Pts Earned
search:		
Opens file for reading	5	
Reads each record of the file	8	
Displays matched records	8	
Properly closes the file	5	
display_names:		
Opens file for reading	4	
Reads each record of the file	8	
Displays all the records of the file	8	
Properly closes the file	5	
new_record:		
Opens file for writing with append	5	
Prompts the user for the data	8	
Tests that the last name is not empty or space	8	
Writes the record with the specified format	8	
Properly closes the file	5	
Program runs as specified	15	
Points earned	100	

Note: If your program fails to compile, you will receive a zero for the assignment.

Comments: