420-200-VA | Assignment 3B | Due by April 2, 2020

Reading and Writing Text Files

1 Purpose

The purpose of this assignment is to refresh your memory on text file input and output processing, effectively repeating one of our most recent lab practices. Specifically:

- Read Text Files Using `Scanner` (Chapter 11)
  - Use `Scanner`'s `useDelimiter` method
  - Read values from lines of comma separated values
- Use `ArrayList<>` of object references (Chapter 9)
- Write Text Files Using `PrintWriter` (Chapter 11)
- Promote hands-on approach to learning by doing
- Build a basis for a series of "running assignments" which continue for the remainder of the course.

2 Requirements

- Although this assignment provides the entire solution code, it won't give you the actual code, so you'll need to actually type in the code, paying attention to the steps and statements involved. Please remember, no copy/paste during job interviews!
- Provide Javadoc documentation for each class
- Complete this assignment in 4 days by Thursday, April 2, 2020

3 Input

A text file named "`./dog_infile.txt`" containing the following lines of text:

```
Lucy, 12 , F , Labrador Retriever
Toby, 1 , M , Alaskan Malamute
Archie, 7 , M , Poodle (Miniature)
Frankie, 9 , M , Staffordshire Bull Terrier
Bear, 8 , m , Dachshund
Milo, 12 , M , German Shepherd
Max, 10 , m , Great Dane
Max, 5 , m , Poodle
Luna, 7 , F , Poodle
```

```
Teddy, 6 , M , Maltese
Duke, 8 , M , French Bulldog
Daisy, 1 , f , Pug
Teddy, 6 , M , Cavalier King Charles Spaniel
Bella, 12 , f , Yorkshire Terrier
Frankie, 12 , m , Jack Russell Terrier
```

Each line in the file represents information for a dog. Each line consists of data separated by commas in this order: dog name, age, gender, and breed.

4 Output

A text file named "./dog_outfile.txt" containing the following lines of text:

```
I'm Lucy, a 12 year old female Labrador Retriever
I'm Toby, a 1 year old male Alaskan Malamute
I'm Archie, a 7 year old male Poodle (Miniature)
I'm Frankie, a 9 year old male Staffordshire Bull Terrier
I'm Bear, a 8 year old male Dachshund
I'm Milo, a 12 year old male German Shepherd
I'm Max, a 10 year old male Great Dane
I'm Max, a 5 year old male Poodle
I'm Luna, a 7 year old female Poodle
I'm Teddy, a 6 year old male Maltese
I'm Duke, a 8 year old male French Bulldog
I'm Daisy, a 1 year old female Pug
I'm Teddy, a 6 year old male Cavalier King Charles Spaniel
I'm Bella, a 12 year old female Yorkshire Terrier
I'm Frankie, a 12 year old male Jack Russell Terrier
```

5 Process

Step 5.1

Create a NetBeans project named DogFileIOApp

Step 5.2

Copy the attached ./dog_infile.txt into your NetBeans project folder

Step 5.3

Add a PetDog class to your DogFileIOApp project:

```
1  public class PetDog
2  {
3     private String name;
4     private int age;
5     private char gender;
6     private String breed;
7
8     @Override
9     public String toString()
10    {
11       return "I'm " + name + ", a " + age + " year old "
12               + ((gender == 'M') ? "male " : "female ") + breed;
13    }
14    // Initializes this new pet dog object, using data values from supplied inputline
15    public PetDog(String inputLine)
16    {
17       // create a Scanner using inputLine
18       Scanner lineScanner = new Scanner(inputLine);
19       // Tell lineScanner that inputLine consists of data separated by commas,
20       // which are each preceded and followed by zero or more spaces
21       lineScanner.useDelimiter("\\s*,\\s*");
22       // extract the data values from the input line
23       this.name   = lineScanner.next();    // read name
24       this.age    = lineScanner.nextInt(); // read age
25       this.gender = lineScanner.next().toUpperCase().charAt(0); // read gender
26       this.breed  = lineScanner.next();    // read breed
27       lineScanner.close(); // close lineScanner
28    }
29    public void pant(){
30       // ...
31    }
32    public void guardHome(){
33       // ...
34    }
35
36    // other members not shown for brevity
37 }
```

Taking an input line of comma separated values as the only parameter, the constructor creates a `Scanner` using `inputLine` on line 18. It then instructs `lineScanner` on line 21 to use the *regular expression* `\s*,\s*` as the delimiter during scanning the input line. Finally, having read the data values, it closes the `lineScanner` object on line 27.

The string "`\\s*,\\s*`" represents the regular expression `\s*,\s*`, meaning zero or more spaces followed by a comma followed by zero or more spaces. The following is an example of a line of comma separated values:

`Max, 10 , m , Great Dane`

Step 5.4

Add a TextFileIO class to your DogFileIOApp project:

```
public class TextFileIO
{
```

```
// reads the lines in the supplied file, converts each line to a pet dog object,
// and adds each dog object to an array list which it finally returns
   public static ArrayList<PetDog> readTextFileToArrayList(String fileName)
                                 throws FileNotFoundException
    {
      ArrayList<PetDog> dogList = new ArrayList<>(); // create storage for dog objects
      File file = new File(fileName);
      Scanner fileScanner = new Scanner(file);       // open the file for input

      while(fileScanner.hasNextLine()) // while there is a next line in the file
      {
        String line = fileScanner.nextLine();     // read a line
        PetDog dog = new PetDog(line);            // turn line into a dog object
        dogList.add(dog);    // add the newly created dog object to our dog list
      }
      fileScanner.close(); // remember to close files when your are done with them
      return dogList;
    }
```

```
// Prints the dog objects in the supplied dog list to a file named fileName
   public static void writeArrayListToTextFile(ArrayList<PetDog> dogList, String fileName)
                       throws FileNotFoundException
   {
      PrintWriter pw = new PrintWriter(fileName); // open the supplied file for output
      for(PetDog d : dogList) // scan the dog list one dog object at a time
      {
         pw.println(d);    // write the cuurent dog object to the file
      }
      pw.close(); // remember to close files when your are done with them
   }
```

```

}
```

Notice that none of the two methods above handles the possible IOExceptions but simply declares that it may throw an IOException, or specifically a FileNotFoundException.

Step 5.5

Add a TextFileIoTestDriver class to your DogFileIOApp project:

```
public class TextFileIoTestDriver
{
   public static void main(String[] args) throws FileNotFoundException
   {
      String inFileName = "./dog_infile.txt"; // must exist in the project folder
      ArrayList<PetDog> dogList = TextFileIO.readTextFileToArrayList(inFileName);

      String outFileName = "./dog_outfile.txt"; // in the project folder
      TextFileIO.writeArrayListToTextFile(dogList, outFileName);

      return;
   }
}
```

Again, notice that method main does not handle the possible IOExceptions that TextFileIO's writeArrayListToTextFile method may throw; instead, it simply declares that it may throw an FileNotFoundException.

In a later assignment, we'll learn how to instruct our methods to handle exceptions directly.

6 Help

Please make sure you ask questions, preferably sharing them on class discussion forums on Lea, if you are not clear about any aspect of this assignment.

7 Evaluation Criteria

Evaluation Criteria

Functionality	Ability to perform as required, producing correct output for any set of input data, Proper implementation of all specified requirements, Efficiency	60%
Robustness	Ability to handle input data of wrong type or invalid value	10%
OOP style	Encapsulating only the necessary data inside objects, Information hiding, Proper use of Java constructs and facilities.	10%
Documentation	Description of purpose of program, Javadoc comment style for all methods and fields, comments on non-trivial steps in all methods	10%
Presentation	Format, clarity, completeness of output, user friendly interface	5%
Code readability	Meaningful identifiers, indentation, spacing, localizing variables	5%