Lab 3 String Processing

This lab will give you practice using many of the Java String library functions. You are required to parse and validate input records from a text file. The name of the input file will be passed to your program using command line arguments. Each line of input will contain a single record containing:

Firstname, Lastname, Gender, Age, Phone Number, Email Address

E.g. Jimmy, Smith, mAlE , 88, (564) 375-9090, hammondmaster@blues.com

- Firstname: Should only contain upper and lower case letters.
- Lastname: Should only contain upper and lower case letters.
- Gender: male or female are the only two values. This field is not case sensitive so upper and lower case letters can be used.
- Age: Must be between 1 and 130.
- Phone Number: (xxx) xxx-xxxx (each x can only be a digit) – just confirm that there are 10 digits.
- Email Address: LettersandDigits@LettersAndDigits.net|com – The email address must start off with a letter, include only letters, digits, periods and the '@'character.

Each of the above fields will be separated by a comma. You are required to:

- Read in each line of input using the Scanner *nextLine()* method. You can assume that the input file will have **no more than 100 records**. You will need to declare an integer variable to keep track of how many records are actually stored in your 1D array of strings (described below). You are free to use an ArrayList if you want to avoid using the extra variable used to track the actual number of array locations used.
- Parse the line of input using the String *split()* method
- Validate each field according to the description in the field list above. Create a different method to validate each field. The method that validates the first name can also be used to validate the last name. If the line of input has an error, immediately output the field and the error. If the field has no errors, format and store the input record fields in an 1D array such that when printed, all the fields will line up in columns.

  You may find the following String and Character methods helpful:

  - charAt()
  - indexOf()
  - trim()
  - format()
  - toUpper()
  - toLower()
  - length()
  - substring()
  - split()

- Character.isDigit()

➢ After all the input has been processed, output the valid input stored in the 1d array to the console. This output should have neatly aligned columns.

Level 1 (10pts): Your program should obtain the filename from the command line argument, use the split function to parse the input, then store the input fields into a formatted string and store each string into the 1D array. Finally, output the array to the console.

Level 2 (15pts): Complete level 1 and validate the Firstname, Lastname and Gender fields. If errors are found, output the errors but do not store records with errors in the array.

Level 3 (20pts): Validate all fields.

Hint:

```
String output = String.format("%-20s%-20s%-10s%10d", "William", "Bryant", "Male", 22 );
String sFirstname = “Helen”;
String sLastname = “Bryant”;
String sGender = “female”;
int age = 9;
String output2 = String.format("%-20s%-20s%-10s%10d", sFirstname, sLastname, sGender );
System.out.println(output);
System.out.println(output2);
```

Produces:

```
William             Bryant              Male              22
Helen               Bryant              female             9
```

The first %-20s means to output the first value (i.e. “Kelvin”) into field of 20 characters that is left justified (the minus sign means left justify and the ‘s’ means that the corresponding value (“Kelvin”) is of type String. %10d is a right justified integer value. For more examples, see:

https://dzone.com/articles/java-string-format-examples