CS 200 Homework Assignment 4
Given: 3 November 2020
Due: 29 November 2020 @ 11:59 PM
100 points available

This assignment involves the following Java topics:

1. User-defined methods
2. Input and Output
3. Exceptions
4. Generics
5. Searching and Sorting

For HW4, you may work in pairs (no groups larger than two) or individually. However, if you work in pairs, I will have higher expectations on your project.

For this assignment you will read in a comma-separated value (CSV) file that lists each US state and county, as well as the number of cases of Lyme Disease that was reported in each county during four separate five-year periods. The CSV file to use is provided in Canvas. You will perform some sorting and searching operations and output some summary data. A HW4 project template file (also provided in Canvas) is provided to get you started.

A CSV file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. A CSV file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields.

The column headers are provided the first line (row) of the file. There is a total of 8 columns and 3194 rows (plus one header row). The fields (columns) are:

1. StateCode,
2. CountyCode,
3. StateName*,
4. CountyName*,
5. ConfirmedCount_1992_1996,
6. ConfirmedCount_1997_2001,
7. ConfirmedCount_2002_2006,
8. ConfirmedCount_2007_2011

* = items that should remain as Strings (the rest should be converted to Integers).

The file has one header row. The first four rows of data (after the header row) look like the following:

```
1,0,Alabama,Alabama,,,,1
1,1,Alabama,Autauga County,,2,,
1,3,Alabama,Baldwin County,4,4,1,1
1,5,Alabama,Barbour County,,,,
```

The HW4 template code will assist you in reading in the CSV file. It replaces the null values with zeroes.

A. In this assignment modify the HW4 template file to:
   - prompt the user to read in a file name using an absolute path. The HW4 project template uses a hard-coded file location for the .csv file. It has incorporated many of the `try..catch` structures needed, but you should feel free to add some as required.
   - ignore the first row in your reading of the file, since it only contains a header row.

- develop the correct collection for reading in the 3194 state-county combinations in the provided .csv data file. The HW4 template provides for an ArrayList of type String, but if you use the ArrayList, you will need to modify it to store some columns as strings and some columns as integers. If you use other types of collections, such as HashMap, Sets, Lists, or Arrays, you will need to modify the code accordingly.

Note: you should not modify the CSV file in any way. I will use a version of this file to test your code.

Next, give the user a choice to search or sort.

```
Do you wish to (1) Search or (2) Sort by a column? (Enter 1 or 2): 2
```

Again, the values entered should be checked to ensure only one of the following values are entered: {1, 2}. If not, an exception should be raised, a message printed to the screen to alert the user, and the user should be re-prompted until a valid value is entered.

B. Develop a generic method for sorting values by any one of the 8 columns and prompt the user which one to sort by as the primary sort column and which one to sort by as the secondary column. For example:

```
1. State Code
2. County Code
3. State Name
4. County Name
5. Confirmed Count (1992-1996)
6. Confirmed Count (1997-2001)
7. Confirmed Count (2002-2006)
8. Confirmed Count (2007-2011)
```

```
Enter the primary sort column (1-8):
```

```
Enter the secondary sort column (1-8):
```

A check should be put in place to ensure the values are only the numbers 1 through 8. An exception should be raised, which gives a message to the user, and then they should be re-prompted until the values are within the specified range of values.

Another check should be put in place to ensure the primary and secondary sort columns are not the same number. If so, an exception should be raised, which gives a message to the user, and then they should be re-prompted for the secondary sort column until it is a different number than supplied by the primary column and a valid number between 1 and 8.

C. It should then allow for a sort on the primary sort column in either *ascending* or *descending* sort order and the secondary column in either *ascending* or *descending* sort order. It should output the results to the screen.

```
Sort primary column in (a)scending or (d)escending order: (a or d)?
```

```
Sort secondary column in (a)scending or (d)escending order: (a or d)?
```

A check should be put in place to ensure the values are only in the following set of values: {a, A, d, D}. An exception should be raised, which gives a message to the user, and then they should be re-prompted until the values are an acceptable value. Instead of a, d, A and D as acceptable inputs, you can use numbers (1 and 2, for example).

D. Another method should be written to output the 4 integer values for a five-year range by either state or county:

- Confirmed Count (1992-1996),
- Confirmed Count (1997-2001),
- Confirmed Count (2002-2006),

- Confirmed Count (2007-2011)

It should be written as a generic method, that will accept either:

- a state code (an Integer) or state name (a String)
- a county code (an Integer) or county name (a String)

You can either prepare one generic method to handle all four cases (state code, state name, county code, and county name) or two generic methods – one to handle either state code or state name and another to handle either county code, or county name. They are to be written as generic methods to handle two types of data.

Be careful when you parse the inputs for the string, since most data fields have a space in them (e.g. Weld County).

It should output the results to the screen.

Some examples:

Example 1: Bad file name.

```
Welcome to Lyme Disease Counts (1992-2011)!

Enter a full pathname and filename for the input file: filename.csv

This filename cannot be found.  Exiting.
```

Example 2: Search by state name.

```
Welcome to Lyme Disease Counts (1992-2011)!

Enter a full pathname and filename for the input file:
C:/Users/public/LymeDisease_9211_county.csv

Reading file…. Done

Do you wish to (1) Search or (2) Sort by a column? (Enter 1 or 2): c

Invalid input.  Enter 1 or 2 only.

Do you wish to (1) Search or (2) Sort by a column? (Enter 1 or 2): 1

Search by:

1. state code
2. state name
3. county code
4. county name

Enter your choice (1-4): 6

Invalid input.  Enter 1-4 only.

Enter your choice (1-4): 1

Search by state code.  Enter a state code: 50

50 is Vermont

Count 92-96: 72
Count 97-01: 103

Count 02-06: 289

Count 07-11: 1538
```

Example 3: Search by county code.

```
Welcome to Lyme Disease Counts (1992-2011)!

Enter a full pathname and filename for the input file:
C:/Users/public/LymeDisease_9211_county.csv

Reading file…. Done

Do you wish to (1) Search or (2) Sort by a column? (Enter 1 or 2): 1

Search by:

1. state code
2. state name
3. county code
4. county name

Enter your choice (1-4): 4

Search by county name.  Enter a county name: Allen County

Count 92-96: 6

Count 97-01: 13

Count 02-06: 6

Count 07-11: 11
```

Example 4: Sort by State Name and Confirmed Count (1997-2001).

```
Welcome to Lyme Disease Counts (1992-2011)!

Enter a full pathname and filename for the input file:
C:/Users/public/LymeDisease_9211_county.csv

Reading file…. Done

Do you wish to (1) Search or (2) Sort by a column? (Enter 1 or 2): 2

1. State Code
2. County Code
3. State Name
4. County Name
5. Confirmed Count (1992-1996)
6. Confirmed Count (1997-2001)
7. Confirmed Count (2002-2006)
8. Confirmed Count (2007-2011)

Enter the primary sort column (1-8): 9

Invalid input.  Enter 1-8 only.
```

```
Enter the primary sort column (1-8): 0

Invalid input.  Enter 1-8 only.

Enter the primary sort column (1-8): 3

Enter the secondary sort column (1-8): 3

Primary and secondary sort column is the same.  Please retry.

Enter the secondary sort column (1-8): 6

Sort primary column in (a)scending or (d)escending order: (a or d)? Q

Invalid input.  Enter a or d only.

Sort primary column in (a)scending or (d)escending order: (a or d)? D

Sort secondary column in (a)scending or (d)escending order: (a or d)? a


Printing results...

56 0     Wyoming     Wyoming           2     1     0     1

56 19    Wyoming     Johnson County    0     1     0     1

56 1     Wyoming     Albany County     2     1     1     0

56 5     Wyoming     Campbell County   3     1     0     0

... (all values are output… this is omitted to save space)

1  119   Alabama     Sumter County     0     0     0     0

1  121   Alabama     Talladega County  0     0     1     0

1  127   Alabama     Walker County     0     0     0     0

1  131   Alabama     Wilcox County     0     0     0     0
```

Example 5: Search by County Name.

```
Welcome to Lyme Disease Counts (1992-2011)!

Enter a full pathname and filename for the input file:
C:/Users/public/LymeDisease_9211_county.csv

Reading file…. Done

Do you wish to (1) Search or (2) Sort by a column? (Enter 1 or 2): 1

Search by:

1. state code
2. state name
3. county code
4. county name

Enter your choice (1-4): 4

Search by county name.
```

```
Enter a county name: Clinton County

Count 92-96: 28

Count 97-01: 33

Count 02-06: 80

Count 07-11: 172
```

Example 6: Sort by Confirmed Count (2007-2011) and County Code.

```
Welcome to Lyme Disease Counts (1992-2011)!

Enter a full pathname and filename for the input file:
C:/Users/public/LymeDisease_9211_county.csv

Reading file…. Done

Do you wish to (1) Search or (2) Sort by a column? (Enter 1 or 2): 2

1. State Code
2. County Code
3. State Name
4. County Name
5. Confirmed Count (1992-1996)
6. Confirmed Count (1997-2001)
7. Confirmed Count (2002-2006)
8. Confirmed Count (2007-2011)

Enter the primary sort column (1-8): 8

Enter the secondary sort column (1-8): 2

Sort primary column in (a)scending or (d)escending order: (a or d)? d

Sort secondary column in (a)scending or (d)escending order: (a or d)? A


Printing results...

9  0     Connecticut       Connecticut             702   576   323   3944

42 29    Pennsylvania      Chester County          893   3580  3580  3342

42 17    Pennsylvania      Bucks County            983   2152  2401  2755

25 17    Massachusetts     Middlesex County        173   469   1496  2585

... (all values are output… this is omitted to save space)

51 620   Virginia          Franklin city           0     0     0     0

51 690   Virginia          Martinsville city       0     0     0     0

51 720   Virginia          Norton city             0     0     0     0

51 730   Virginia          Petersburg city         1     2     1     0
```

What to submit:

- A single BlueJ project using the following naming convention: *<last name1><last name2>HW4* for submissions from a pair of students or: <*last name*>HW4 for individuals. A deduction will be made if you submit only a pdf file or other types of files.
- In your comments section, indicate which person did what part of the assignment in 2-3 sentences (this is not needed for individual submissions).
- A narrated short video (or link to the video on YouTube or other content sharing service) showing your code and illustrating your code running. In your video include the examples given above in your demo. A deduction will be made for not including a video.
- The video and the BlueJ project can be zipped together in a single file or submitted as separate files.
- For people working in pairs, only one submission is necessary for the two of you.

Grading:

As with previous homework assignments, make appropriate comments in your code – they do not need to be excessive but should include a clear, concise understanding of the code you have written. Also, include a header for each class file. There will be a deduction of up to 5 points each if these are not included.

Submit your code and videos in the format provided in "what to submit" – a deduction of up to 10 points will be made if these are not followed.

Use appropriate variable names that clearly illustrate their purpose. For counting variables, you can use single letters, but all others should be descriptive. A deduction of up to 5 points will be made for using inappropriate variable names.

Coding should follow the instructions above. While the rubric for this assignment does not specifically allocate points for each method, the breakdown is *roughly* as follows:

- Part A – Developing the collection to accept CSV files (15 points)
- Part B – Generic method for sort (20 points)
- Part C – Primary and secondary sort (20 points)
- Part D – Search (30 points)
- Output formatted as given in the examples (5 points)
- Exceptions to handle incorrect conditions (10 points)

Pay attention to the deadline for this assignment. Penalties do apply for late submissions.