CSC 230 – Project #1
Spring 2015

Last modified: 2/7/15 9:43 PM

Goals: Use searching and sorting techniques to process a list of US zip codes to answer several questions regarding the postal zip code system.

Part 1: Project Summary (Due: 3pm Feb. 20th, 2015; Drop dead date: 3pm Feb. 23rd, 2015)

Provided via my web site (http://s3.amazonaws.com/depasquale/datasets/zipcodes.txt) is a list of all US postal zip codes in a plain text file. The format of the file is as follows. Each line of data represents one zip code location. A comma separates data on each line, and each data value is surrounded by double quotes.

Data is organized on each line as follows: the zip code, the latitude of the postal location, the longitude of the postal location, the name of the postal location (generally the city or town name), the state of the postal location (including US territories), the county of the postal location and the type of postal location. Note that location names, county names, and postal location types may be composed of multiple words.

A sample from the input file is:

```
"00646","+18.436060","-066.281954","DORADO","PR","DORADO","STANDARD"
"00647","+17.969594","-066.939754","ENSENADA","PR","GUANICA","STANDARD"
"00650","+18.360125","-066.562311","FLORIDA","PR","FLORIDA","STANDARD"
"00652","+18.457254","-066.603358","GARROCHALES","PR","ARECIBO","STANDARD"
"00653","+17.972468","-066.898661","GUANICA","PR","GUANICA","STANDARD"
"00656","+18.023280","-066.786909","GUAYANILLA","PR","GUAYANILLA","STANDARD"
```

Create a **`ZipCodeLocation`** class that contains all of the above information. Do not store the latitude and longitude values as **`String`** objects, but rather doubles. This may require using precision analysis, as shown in your textbook.

Next, write an **`Application`** class that contains a **`main`** method that calls a method named **`readData`** to read, store, and return the data in an array of **`Comparable`** objects. If while reading the input data, you come across a line of input that contains a missing or blank data field, the **`readData`** method should write an informative message to **`System.err`** that lists only the zipcode corresponding to the bad line of data (assuming the zip code is not missing).

Your code that reads data from the input file (use a **`java.util.Scanner`** object) should catch any of the exceptions being thrown by the Scanner (see the API for details on the exceptions being thrown by the **`nextDouble`**, **`nextInt`**, **`hasNextDobule`**, **`hasNextInt`**, **`hasNext`**, and **`next`** methods (and others that you choose to use). If an exception is thrown, provide a descriptive message to the **`System.err`** stream informing the user of the problem, discard the current line, and continue to read the input file from the next line (as if nothing happened). Thus you'll want to implement a try-catch statement around the reading code, perhaps with a finally clause (perhaps not…) within the body of the readData method.

Finally, have your **`main`** method call other methods which produce output that answers the following questions (in this order). Your output should be written to an output file named **`zipout.txt`**. Catch and handle any possible exceptions that could come from opening, writing

to, and closing the output file. Each section that attempts to provide answers for the questions below should be clearly delineated in the output. (See below for an example.)

Tasks/Questions for your program to do/answer

Your output should be properly labeled so that I know which questions you are answering.

- List the towns in Rhode Island (NJ) in ascending alphabetical order. List only each town once (unique listing). (Hint: You can use a sorting algorithm if you are familiar with them, or see the **`java.util.Arrays`** API class…) Hard-coding the printing of the list is not acceptable, this must be completed programmatically.
- Provide a unique list (no duplicates) of all of the counties to which the USPS delivers in New Jersey. Sorting of the list is not necessary. Hard-coding the printing of the list is not acceptable, this must be completed programmatically.
- List all of the locations (city, state), which have the same latitude and longitude values but different zip codes. Print the name of the location, the la/lon value and each zip code in that location.
- What are all of the different types of postal code types (last column) and how many of each exists? (Determine this programmatically.)
- How many total counties does the USPS deliver to?
- List all of the zip codes whose postal location are either named Springfield or have Springfield in their name (e.g. West Springfield).

The input file is named **`zipcodes.txt`** and is available on my server for download. DO NOT INCLUDE THIS FILE IN THE JAR YOU SUBMIT TO CANVAS.

Sample output (incomplete and using hypothetical results – do not use for program output validation):

```
---------------------------------------------------------------
Types and frequency of postal code types:
---------------------------------------------------------------
STANDARD - 145
PO BOX ONLY - 45
UNIQUE - 7

---------------------------------------------------------------
Total number of counties that the USPS delivers to:
---------------------------------------------------------------
57

---------------------------------------------------------------
List of all zip codes whose postal location are either named
Springfield or have Springfield in the name:
---------------------------------------------------------------
11413
13333
13468
...
```

Part 2: Ant Requirements

Create and fully test an ANT build file (named build.xml). The build file should contain the following targets:

1. **compile** - compiles the project source code
2. **clean** – removes all of the .class files, back up files (ends with ~, eg. Driver.java~), the META-INF directory and its all of it's contents (if any), and the output file from the build area. This target or its underlying tasks should not fail if the file or directory is not present (output file, META-INF, etc.) META-INF is automatically created when you create a jar file. After unjarring your file, I use clean to delete this directory and its contents.
3. **author** – prints the name of the program's author to the screen

Part 3: Specifics for the delivery of this project:

To submit your project, jar up only the source code file(s) and your ANT build file and submit it to Canvas. Do not include the input file, the output file, or any other files of any type.

The task of organizing and creating the jar file for each deliverable will be up to you. You should always check your work, as this file is what I will be grading. You want to ensure that only the files I request are contained in the jar file.

Be sure to fully check the operation of the Ant build file as well, which may include moving your jar file to a clean (empty) directory, unjarring it and performing the ant commands as I will be doing.

You may submit the project any number of times until the due date. Only the last submission will be downloaded, examined, and graded by me.

Late points will be deducted for projects turned in starting 1 minute after the due date (that's why they are called due dates!) Following the drop-dead date for a project, no solutions will be graded for the project. At such a point in time, you will receive a zero grade for the work if you have not uploaded a solution to Canvas.

At a bare minimum, your program should compile. I expect that in this class you are minimally capable of delivering a barely functional program that can be compiled using the command line tools and/or Ant. Failure to deliver a compliable program will result in an automatic 50 score and further analysis of your submission will cease.

Part 4: Advice and Disclaimer

Plan your schoolwork and life accordingly. Many students don't adequately plan their work schedules and attempt to finish programs at the last minute. Doing so usually introduces bugs/problems into the solution. Consider how much time you will require to adequately test your solution for boundary cases and bugs. Failure to plan on your part does not constitute an emergency on my part!

I do not give individual extensions for projects, and rarely give class extensions for work. You have nearly a week and a half to complete this project. I suggest you start your design as soon as possible. Consider what you need to build to solve the problem at hand. Programming involves designing a solution, implementing your solution, and testing your solution.

Part 5: Do You Need Help?

Are you stuck? Ask yourself if you have planned the design of your project. Have you asked questions in class? Did you attend the lectures, and labs? Have you come to see me? Like a textbook or web site, I am a resource that I expect you to use throughout the course of the semester. If you are stuck with your project, I strongly suggest you make an appointment to come see me in person. Generally, I can offer some guidance or help to get you back on the path.

When you come to see me, plan on bringing your source code to me on a memory stick or CD-Rom. Do not email me your source code, I do not provide email-based help for coursework. I prefer that you make time to visit me in my office where we can sit down and discuss your source code together. I can ask you questions and you can tell me why you coded something the way you did. We can use my whiteboard to draw large diagrams and talk about good solutions and designs. This is not something that can be done easily through email.

Part 6: Code Commenting and Formatting (Indenting) Requirements

Please refer to the separate document provided on Canvas for the details of the commenting and formatting guidelines. Note that they are a part of the grading rubric for this assignment.

CSC 230 – Programming Project #1 Grading Sheet

Implementation: (as elaborated in the spec)	
• Input from plain text ASCII file (obtained through scanning URL) (-5)	
• Data records stored as ZipCodeLocation objects. (-5)	
• Latitude and longitude values stored as doubles (-5)	
• ZipCodeLocation objects implement the java.lang.Comparable interface (-5)	
• Bad data lines produce output listing zip code to System.err (-5)	
• Program uses sorting algorithms (provided) to answer questions about data set	
• Output produced to zipcodes.txt file (-5)	
• List the towns in Rhode Island (RI) in ascending alphabetical order. (-10)	
• Provide a unique list of all counties in NJ that the USPS delivers to. (-10)	
• List all locations that have the same lat/lon but different zip codes. (-10)	
• List all of the different types of postal code types. (-5)	
• Provide the total number of counties to which the USPS delivers. (-10)	
• List all zip codes with "Springfield" in the postal location. (-5)	
• Exceptions caught and handled as specified in project specification document (-5)	
Source Code Formatting and Commenting: (see separate spec)	
• Code commenting includes file-level comments for each file, method comments for each method, and variable comments for each instance variable. (-10)	
• Follows some type of consistent formatting and indenting pattern (-5)	
Late Penalty (if applicable)	
Notes	