CMSC 206: Data Structures
Assignment #2: An Array of Zip Codes
due on Gradescope by the beginning of class on Tuesday, February 13, 2018

Purpose: Write a complete Java program to repeatedly input a zip code from the user. In response, your program will output the name of the town and the state to which it belongs. In case the zip code doesn't exist, it tells the user about it (see example interaction below).

The dataset of all the zip codes in the United States is available in the file *uszipcodes.csv* as posted on the syllabus page. A smaller file (of the same format), named *testZip.csv*, is posted as well, in case you want to debug on a smaller dataset. Here are the details of the data file's format:

- The first line of the data file is a bunch of comma-separated values in the format shown below:

  <*number of zip codes*>,zip,city,state,population,males,females

  The first part of the line is an integer giving you the number of zip codes stored in the file. The rest of the line contains column headers for the file. You will ignore the rest of this line in this assignment, but it may be interesting if you wish to look at the data yourself.

- The rest of the data file contains entries (one line for each zip code) in the following format:

  <*zip code*>,<*town name*>,<*state code*>,<*population*>,<*# of males*>,<*# of females*>

  In other words, the line contains the 5-digit zip code, then the town name, then the two-letter state abbreviation, then the population (as an integer), then the number of males in the zip code, and finally the number of females in the zip code. Each field is separated by one comma. Note that some fields may be left out, and town names may have spaces in them.

- Example entries:

```
56315,Brandon,MN,1591,829,762
56316,Brooten,MN,1580,815,765
56317,Buckman,MN,,,
56318,Burtrum,MN,1238,653,585
56319,Carlos,MN,1321,677,644
56320,Cold Spring,MN,8080,4050,4030
```

  The first line is for the zip code 56315 which belongs to the town of Branden, MN that has a population of 1591 of which 829 are males and 762 are females. The zip code 56317 has no population recorded, and the zip code 56320 has a town name with two words.

In this assignment, you will ignore the population numbers, and store only the zip code, the town and the state.

Your tasks:

1. Create a new project for this homework assignment. (In general, all homework assignments should be in their own Eclipse project.)

2. Create a class called `Place` to model each zip code to contain the following data fields: zip code, town, state. In order to support data hiding and encapsulation, these fields should be labeled `private`. You will thus need a constructor and several methods in the `Place` class to set up the fields and to access them.

3. Download the file *ZipParseException.java* from the syllabus page. Put it in your project *src* folder. Read this file, as you'll be using it in your work.

4. Write a separate class `LookupZip` that will contain several `public static` methods to implement the main part of the assignment:

```
/** Parses one line of input by creating a Place that
 *  denotes the information in the given line
 *  @param lineNumber The line number of this line
 *  @param line One line from the zipcodes file
 *  @return A Place that contains the relevant information
 *  (zip code, town, state) from that line
 */
public static Place parseLine(int lineNumber, String line)
  throws ZipParseException

/** Reads a zipcodes file, parsing every line
 *  @param filename The name of the zipcodes file
 *  @return The array of Places representing all the
 *  data in the file.
 */
public static Place[] readZipCodes(String filename)
  throws FileNotFoundException, ZipParseException

/** Find a Place with a given zip code
 *  @param zip The zip code (as a String) to look up
 *  @return A place that matches the given zip code,
 *  or null if no such place exists.
 */
public static Place lookupZip(Place[] places, String zip)
```

   Write these methods. They should be able to handle any sort of erroneous input; if something is wrong in the file you're reading, throw a `ZipParseException`. (No

more than 20% of the grade on this assignment will be about erroneous files, so don't let error handling stop you from finishing the other parts.) Note that `readZipCodes` should call `parseLine` internally.

5. Write JUnit unit tests for the methods `parseLine` and `lookupZip`, in a separate `LookupZipTest` class. Include at least 5 tests (each in a separate `@Test` method) for each method. (That's a total of at least 10 tests.)

6. Write a `main` method that ties it all together. The automatic testing will include only the methods above, so you do not have to get the input/output correct character-for-character here. However, your program must give the user the opportunity to query multiple zip codes.

   Here is an example session:

```
What zip code should I look up? 19010
The zip code 19010 belongs to Bryn Mawr, PA
Do you want me to search again? yes

What zip code should I look up? 99400
The zip code 99400 does not exist.
Do you want me to search again? yes

What zip code should I look up? 91729
The zip code 91729 belongs to Rancho Cucamonga, CA
Do you want me to search again? no

Good Bye!
```

7. Write up your reflections, answering the following questions in a *reflections.txt* file:

   a. How long did this assignment take you?
   b. What challenged you the most in this assignment?
   c. Whom did you work with on this assignment?
   d. What resources did you consult while doing this assignment?
   e. Do you have any feedback to offer about this assignment?
   f. Any other comments or questions?

8. Submit your work on Gradescope.