## Lab#3: Assignment#3

In this lab, you will work on your Assignment#3 with guidance from this handout. This will enable you towards completion of the assignment and, in the process, you will learn how to do incremental program development.

## Version 0: Getting input

The first step in any programming project is to create the project, and write the smallest program that does something. In this case, you would focus on creating a small test input file, and ensuring that your program is able to read from it.

Create a small test file for input: Go ahead and copy the data file in your project folder (create a Data folder and then place the file in it). Next, edit the file to extract the first 10-15 lines and save it as a new file, testZips.txt in the same Data folder. Edit line#1 of the testZips.txt file to reflect the number of data lines in the file.

In your program's main class, create a method called `readData()` as follows:

```
public static void readData() {

   // Open an input stream to the data file
   // Read the first line from it and extract
   // the number of entries in the file (print this out)
   // Read each line, and print it out in the console

}
```

`readData()` will be responsible for reading the data file. Make sure all the variables needed by `readData()` are defined inside the method. It should have the structure shown above.

Go ahead and complete this step. Once done, answer questions 1, 2 and 3 on the Lab Report.

## Version 0.1: Creating the Place class

Next, create the `Place` class. It should have data fields for the zip code, the name of the town, and the state. It should have a constructor (arguments: zip, town, state), and accessors for each field. Also, create a `toString()` method so it returns strings of the form:

```
Bryn Mawr, PA 19010
```

Next, modify the `readData()` method to parse each input line of data into pieces, create a `Place` object using them, and then printing out the object, as show below:

```
public static void readData() {

   // Open an input stream to the data file
   // Read the first line from it and
   // extract the number of entries in the file (print this out)
   // Read each line, and print it out in the console
      Place p;
      // Parse each line into pieces
      p = new Place(…);
      // Output the place's info
      System.out.println(p);

}
```

Go ahead and complete this step. Once done answer question 4 on the Lab Report.

## Version 0.2: Storing data in an array

In your main class, define an array of place objects as shown below:

```
public static Place[] places;
```

Modify the `readData()` method to (1) Create the places array to contain N place objects, (2) put all the objects into the places array. Once done, comment out the print statements from `readData()` (now that we know they work). Add the following loop to your `main()` method:

```
for (int i=0; i < places.length; i++) {
   System.out.println(places[i]);
}
```

Make sure that the program is able to read the input, parse it, store the place objects in the array, and then print it out.

Once done, answer questions 5 on the Lab Report.

## Version 0.3: User Interaction

Now, your program is almost ready for taking in user requests. Go ahead and build the following interaction in your `main()` method:

```
do {
   // Get input from user, make sure it isn't empty or invalid
   // search, do not do anything here yet
   // output results of search
   // Ask user if they want to do it again
} while (// user continues to do it again);
```

Implement the set of interactions needed (without actually doing the search. Just print out a message saying "Now searching for …" along with the zip code that was input.

Once done, answer questions 6 on the Lab Report.

## Version 0.5: Doing the search

Now you are ready for writing the `search()` method. It will be in the main class and will have the following header:

```
public static int search(Place[] p, String zip) { … }
```

It should return the index in the array of places if the zip code is found in the array, `p`, -1 otherwise. A simple algorithm for doing this is:

```
for each element in the array
   if the element is what you are looking for, return its index
return -1
```

That is, just run through the array from start to end. Return the index when you find what you're looking for. If the loop terminates, it implies you looked at everything in the array and did not find it. Return -1 in that case.

Go ahead and code this. Define the `equals()` method in Place class. It should take a zip code as its parameter and return true if the object's zip code is the same as the parameter, false otherwise. Insert the call to the `search()` method in your main() method. Update the `main()` method to printout the results dialog. Run the program several times, looking for zip codes that are present (including the first and the last one in the data file), as well as several that are not present.

Once done answer question 7 on the Lab Report.

## Version 0.9: Using Real Data

Now, it is time to test your program on the full data file. Place it in the Data folder, modify the program to use this data file, and test the program. Again, run it multiple times, and ensure that there are no remaining issues.

Your program is now complete. You may call it Version 1.0!