

# Homework 12

## Input / Output

**Do this homework for attendance at the eleventh laboratory session**

The following exercises are a tutorial on learning how to do input and output of data from files, and how to do tokenizing to extract data from strings.

The data you will be using for this exercise are stored in the `zip.txt` file. In order to have it download on your hard disk the file `zip.zip` and unzip it.

### Exercise 1

Let consider the file `zip.txt` here included. It contains data about some towns and counties in the United States. Each line of the file contains the following information:

`zip code, Town Name, State, Phone Area Code, Dummy, County, Time Zone, Dummy, Dummy, County Population`  
where the fields labeled, `Dummy`, is information we will not be using in this exercise. Here is an example line from the file:

```
02138,Cambridge,MA,617,0472130681980,Middlesex,Eastern,40,2.63,1367000
```

Cambridge, MA has seven Zip Codes and thus there are seven entries in the data file.

1. Write a Java application that reads all the data in the file and stores it in a list implemented as an array. After reading the

data file, your program should print out the number of entries in the list. Ignore the Dummy fields.

2. Add to your program a new facility described in the followings:

A user enters the name of a town and a state. In response, the program should search the list for the name of the town (the first occurrence only) and state, and then print out all the information it knows about it, in the following format:

```
Query: Cambridge, MA
```

```
Found...
```

```
Town: Cambridge, MA, 02138 (Area Code: 617, Time Zone: Eastern)
```

```
County: Middlesex, population 1367000
```

Of course, a user may misspell the name of a town or enter the name of a town that does not exist (i.e. contained in your database). In that case, just print out a message. The user may also use lower case where upper case is needed or vice versa. This is not an error! You should find the town, too. You are only printing out the first occurrence of the town queried.

*Hint.* The `StringTokenizer` class can be used to separate tokens based on a provided token separation character. That is:

```
StringTokenizer tokens=new StringTokenizer(some_String);
```

will separate tokens by assuming each token is separated by a space (' '). You can, instead issue the command:

```
StringTokenizer tokens=new StringTokenizer(some_String,",");
```

to separate tokens using a comma as a separating character.

In order to navigate in a string you may use `hasMoreTokens()` and `nextToken()` as in the following example:

```
StringTokenizer st = new StringTokenizer("this is a test");
while (st.hasMoreTokens()) {
    System.out.println(st.nextToken());
}
```

prints the following output:

```
this
is
a
test
```

**2.** You will also need to make use of the `trim()` method of the `String` class. Given a string:

```
String s1 = "    Java    ";
s1 = s1.trim();
```

will trim the spaces from the string. Thus `s1` will now contain the value:

```
s1 = "Java"
```