Joanna Klukowska
joannakl@cs.nyu.edu

Programming Project 1: NYC Trees

Due date: February 10, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own. Any sharing or copying of assignments will be considered cheating. If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own).

You are responsible for every line in your program: you need to know what it does and why.

In this project you will provide a tool for extraction of certain type of information about street trees in New York City. The Department of Parks and Recreation performs a census of trees growing on the streets of New York City. The course website has a listing of the files with the most recent data for Manhattan, The Bronx, Brooklyn and Queens (sorry, but the data for Staten Island does not seem to be available in text format). Your program should run with one of those files as the input and extract some interesting information (details below) from it: we'll look for most/least popular species of NYC tree, most/least green zip code, largest tree, etc. All output data should be written to an output file.

The program that you write has to be command line based (no graphical interface) and it should not be interactive (do not ask user for anything).

Objectives

The goal of this programming project is for you to master (or at least get practice on) the following tasks:

- working with multi-file programs
- reading data from input files
- writing data to output files
- using command line arguments
- working with large data sets
- using ArrayList class
- writing classes

Most, if not all, of the skills that you need to complete this project are based on the material covered in cs101. But there might be certain topics that your section did not cover at all, or that you did not get much practice on. Make sure to ask questions during recitations, in class and on Piazza.

Start early!

The Program Input and Output

Input File Your program is given the name of the input text file on its command line (the first argument). The input file has the following structure:

- the first line contains column headings (your program can simply ignore it)
- all remaining lines contain 9 **comma separated entries** of data (your program may not need all of them, but they are all in the file)
  - tree ID
  - street name (where the tree is growing)
  - cross street #1
  - cross street #2

Joanna Klukowska
joannakl@cs.nyu.edu

  - tree condition (an integer)
  - diameter
  - species code
  - borough name
  - zip code

  (note that some of the entries may contain multiple words, for example, "EAST HOUSTON STREET" is a name of a street for some of the trees).

If a line in the input file does not contain correct number of comma separated entries, it should be ignored. The program should continue to the next line. This means that the code that parses the input file needs to handle incomplete lines.

If the filename is omitted from the command line, it is an error. The program should display an error message and terminate. The error message should indicate what went wrong.

If the filename is given but the file does not exist or cannot be opened for reading by the program, for any reason, it is an error. The program should display an error message and terminate. The error message should indicate what went wrong.

Your program is NOT ALLOWED to hardcode the input filename in its own code. It is up to the user of the program to specify the name of the input file. Your program should not modify the name of the user-specified file (do not append anything to the name).

Your program may not read the input file more than once.

Output File Your program should produce an output text file. Its name should be constructed based on the input file name: it should match the input file name without the extension (if any extension is peresent in the input file name) and it should have `.out` extension. For example

- if the input file name is `Manhattan_trees.csv`, the output file should be named `Manhattan_trees.out`,
- if the input file name is `BrooklynTreeCensus`, the output file should be named `BrooklynTreeCensus.out`,
- if the input file name is `Queens.txt`, the output file should be named `Queens.out`.

If a file with the given filename exists already, your program should overwrite it without any warning. If the program cannot create a file with the given name, for any reason, it should print an error message and terminate. The error message should indicate what went wrong.

Your program is NOT ALLOWED to hardcode the input filename in its own code.

User Input This program is not interactive. It should not prompt or expect any input from the user.

Computational Task

Once the program reads the data from the input file, it should perform the following tasks. (The program can perform these tasks in whatever order you wish, but the output file should contain the data in the order and format specified below.)

For some of the tasks, your program will need to convert the species code of the tree to its full name. The course website provides a text file called `species_list.txt`. Each of the lines in the file contains a single species code (first word on a line) followed by its full name (often consisting of multiple words). Your program should assume that this file exists in the current working directory and is named exactly `species_list.txt` (its name should be hardcoded into your program).

Note1: the sample outputs are made up and dot not correspond to any particular input file.

Note2: the input files may contain incorrect information - it is not the job of your program to detect that.

Note3: your data can be printed using upper case, lower case or mixed letters.

Joanna Klukowska
joannakl@cs.nyu.edu

Task 1 Find the three most frequently occurring trees. Their names and counts should be printed to the output file in the following format.

```
Most popular trees:

    ash, white 10385
    ginkgo 8452
    elm, siberian 6490
```

If there is a tie, the names of all the trees with the same count should be provided, for example:

```
Most popular trees:

    ash, white 10385
    ginkgo 8452
    maple, amur 6490
    elm, siberian 6490
```

Task 2 Find the three least frequently occurring trees. The species of the trees from the `species_list.txt` that are not found in the input file should be ignored (this means that the counts of the least popular trees should not be zero). Their names and counts should be printed to the output file in the following format.

```
Least popular trees:

    ash, white 10
    ginkgo 9
    elm, siberian 1
```

If there is a tie, the names of all the trees with the same count should be provided.

Task 3 Find the three zip codes with the largest number of trees. If there is a tie, all zip codes with the same count should be listed. The zip codes and the counts should be printed in the following format:

```
Most green ZIP codes:

    10025 985
    10003 870
    10012 562
```

Task 4 Find the three zip codes with the smallest number of trees. If there is a tie, all zip codes with the same count should be listed. The zip codes and the counts should be printed in the following format:

```
Least green ZIP codes:

    10021 20
    10013 15
    10022 13
```

Task 5 Find the thickest tree. If there is a tie, all trees with the same (largest) diameter should be listed. The tree information should be printed in the following format:

```
The largest tree:

    Linden, silver, 90 inches in diameter
    Madison Avenue (East 81 Street, East 82 Street)
    10028
```

The first line contains the tree's name and its diameter. The second line contains the street on which the tree grows followed by the names of the two cross streets in parenthesis). The third line contains the zip code in which it is located.

Given the above sample outputs, the output file should contain:

Joanna Klukowska
joannakl@cs.nyu.edu

```
Most popular trees:

    ash, white 10385
    ginkgo 8452
    maple, amur 6490
    elm, siberian 6490

Least popular trees:

    ash, white 10
    ginkgo 9
    elm, siberian 1

Most green ZIP codes:

    10025 985
    10003 870
    10012 562

Least green ZIP codes:

    10021 20
    10013 15
    10022 13

The largest tree:

    Linden, silver, 90 inches in diameter
    Madison Avenue (East 81 Street, East 82 Street)
    10028
```

Program Design

Your program must contain three classes:

- `Tree` class to represent a single tree. An object of class `Tree` contains all the information about a single tree.
- `TreeList` class to represent all the Tree objects in a single container. `TreeList` class should store all the `Tree` objects in an ArrayList of trees (depending on your design, you may use more than one such ArrayList). This class should provide methods that return the results needed by the five tasks described in the previous section.
- `TreeInfo` class that is the runnable program containing the `main()` method. This class is responsible for parsing the command line argument, reading the input file, creating the `TreeList` object based on the input file, calling all the methods of the `TreeList` object needed for completion of the five tasks, and writing to the output file.

Programming Rules

You must document all your code using Javadoc. Your class documentation needs to provide a description of what it is used for and the name of its author. Your methods need to have description, specification of parameters, return values, exceptions thrown and any assumptions that they are making.

Class's data fields and methods should not be declared `static` unless they are to be shared by all instances of the class or provide access to such data.

You may use `ArrayList` and `String` classes and any methods that are provided in them.

You may use any classes for reading the input from the file and writing the output to the file.

You may use any exception related classes (if you wish).

You may use the sort/search methods provided in the Arrays and/or Collections class (but this is going to be possible only if you design your `Tree` class appropriately).

Joanna Klukowska
joannakl@cs.nyu.edu

Working on This Assignment

You should start right away! This program does not require you to write much code (well, more than you are used to from cs101, but less than future assignments), but it will take some time.

You should modularize your design so that you can test it regularly:

- Start with a program that reads in the input file. Make sure that this works. (This can be tested easily, by simply writing the data back to an output file without any changes.)
- Write a class to represent a tree. Make sure that this works. Think about all methods that this class may need to provide.
- Write a class that represents the list of trees.
- Write the main() function that combines the above pieces and produces the output file.
- Test, test, test, test. Use different input files. Try to break your program.
- Submit your code.

You should backup your code after each time you spend some time working on it. Save it to a flash drive, email it to yourself, upload it to your Google drive, anything that gives you a second (or maybe third copy). Computers tend to break just a few days or even hours before the due date - make sure that you have working code if that happens.

Grading

If your program does not compile or crashes (almost) every time it is ran, you will get a zero on the assignment.

If the program does not adhere to the specification, the grade will be low and will depend on how easy it is to figure out what the program is doing.

15 points	design, implementation and correctness of the Tree class
35 points	design, implementation and correctness of the TreeList class
25 points	design, implementation and correctness of the TreeInfo class
25 points	proper documentation

How and What to Submit

Your should submit all your source code files (the ones with .java extensions only) in a single **zip** file to NYU Classes. You should have three different files: `Tree.java`, `TreeList.java` and `TreeInfo.java` (make sure that these are the names of the files).