COP 2551 – Introduction to OOP
Program #4
Due: 31 July 2007

Using NetBeans 5.5, you are to write a Java program using OOP principles to accommodate the following functionality

Assignment #1

Objectives:

Provide student with experience building arrays of objects
Provide student with opportunity in doing file input output.
Provide student with exercises in learning UML
Provide student with exercises in Javadoc and its various formats
Provide student with exercises in searching and traversing the array of objects.

Functionality:

Using States.Spring2007.txt, you are to do the following:

1. **Build array.** Using States.Spring2007.txt, you are to build an array of objects. You are to develop a class named **State** and create 50 objects of **type (class) State** – one object for each record (line) from the input file.

(You may cut off the first few lines of information. These are for you to see the relative position and length of each attribute. You are **not** to alter the data, however, in any way.) Each State object will have six properties defined individually as ints, Strings, or whathaveyou, as appropriate for each State object. Hint: you cannot use StringTokenizer. You will need subString to get to the individual attributes.

2. **Display files Requirement**. From main() you are to write code to print the array of objects. Write these to your screen via displaying; copy them also to a file called States.print.txt.

On the displayed output (screen), you are to have a column header, followed by a blank line, followed by single spaced (one line per object) output. For the file output, only write the records.

3. **Scan and total requirement. Using the array of State objects, you are to** examine each State object and accumulate data. At the end of this scan, you are to print out a header that says Population By Region (left justified) and underneath this, you are to print the number of states and the average population of states in that region. (See data: regions are numbered 1 through 6). Your format for these detail lines should appear as:

Population By Region

New England 6 5,321,123

(note the spacing; use tabs)

At the end of printing these lines, you are to print:

Population of the United States:

StateCount 50 301,222,333 (or whatever you add up to)

4. Search Requirement. Using input transaction file, States.Trans, you are to search through your array of state objects again and identify 'hits' or 'no hits.' (match or no match…)

Read an input token and search through the array of state objects. If a hit is encountered, you are to display:

Hit for input: <input item> <advance to next line>
State Particulars are: <State name, State pop, State abbreviation, State Region>
Ensure these are spread out via tabs or suitable spacing…

There is to be a header **prior** to any displaying of course. It should read: Search Engine Results (left justified). Skip a line (blank line) before you start printing the Hit for input line and others. These outputs (detail results from the search) are to be single spaced.

Please note that there will be a number of hits, as a number of the input search arguments are misspelled on purpose. For those input arguments for which there is no hit, you are to display:

No Hit for Input: <input item> <advance to next line – that is, leave a blank line in between this output line and whatever might follow.

At the end of searching, you are to display the number of successful searches and the number of unsuccessful searches according to the format:

<blank line>
Summary Statistics:
Number of Successful Searches: <an integer>
Number of Unsuccessful Searches: <an integer>

Ensure there are spaces between your text and the integers you display.

UML

You are to include a UML class diagram. You may use Word or Power Point. **No other technology may be used!**

Use the examples in your 2551 text. Each class listed in your UML diagram must have attributes listed (name, type). Methods must be shown with visibility indicator, and number /type of arguments plus the return type. Connect all classes (label the associations). Use UML format not Java. See previous lectures and your text for examples.

'Drag' your UML design file into your P4 subfolder within your COP2551 desktop folder. It will be included in the zip file to me. All other files should also be included in this folder which also must include the initial States file too.

Javadoc

All programming is to be accompanied by appropriate Javadoc.

ALL methods are to have Javadoc comments preceding them (in your source code). Appropriate documentation for methods consists of a short description (single sentence) plus any parameters and any return types specified via @params and @return.

Appropriate documentation for **classes** includes several sentences describing the purpose of the **class**. Include @author and any other documentation that assists in documenting that particular class. (Of course, objects related to this class should be described via accompanying UML diagrams.)

Javadoc generation:
From the Project window pane, right click on the projectname at the top of the pane.
Select Properties (last choice in the drop down).
Select Documenting
Select Document Additional Tags – Author
Click OK
All Javadoc will be generated and moved to your program folder. (What a deal!)

Do this at the very end of your programming effort so as to ensure all comments are captured.

You are to zip all files in your P4 as expected and Send them to be via Digital Dropbox using the same naming conventions as in P4.