

# COP 2551 – Introduction to OOP

## Program #4

**Due: Wednesday, 17 Nov 2010 start of class**  
**Drop dead: Monday, 22 Nov 2010 start of class.**

Using NetBeans 6.8, you are to write a Java program using OOP principles to accommodate the following functionality

### Assignment #4

#### Objectives:

- Provide student with experience building arrays of objects
- Provide student with opportunity in doing file input and 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 the external file States.txt, you are to do the following:

##### 1. Build an array of State objects.

You are to develop a class named State from an external file, BigStates.Fall2010, and create as many objects of this type – one object for each record (line) from the input file. *Allocate your array size to the number of input strings, which I believe is 33.* ~~While you may look at the file and its contents, do not 'read in' the exact number of states as lines, as I may test your program with a different file with a different number of states:~~

You are **not** to alter the data, however, in any way.) Each state object will have properties as shown and defined individually as Strings, ints, or whathaveyou, as appropriate for each state object. Hint: you may use substring method in class String to parse as expected. Please note that the first several lines provides the layout of the data below. When you Save Target As to download this state file, you may eliminate those first lines. But you need the layout of information in your program in order to parse. Thus, only save into your project area a download of the data lines.

##### 2. Display files Requirement.

From main() you are to write code to display the array of objects to the screen – you must use toString(). This is to include a nice looking header spanning (spread out) the display line followed by nicely spaced columns of state attributes aligned under their respective header. Text data is to be left justified; numeric data, always right justified with commas as appropriate.

3. **Copy the array of objects to an external file** to BigStatesOut.Fall2010. See slides for example. You will need FileWriter and appropriate methods as found in the slides.

For the file output, only write the records. No header line. So the individual lines should look like what you display in #2 above – nicely spaced out; no header.

4. **Scan and Total Populations Requirement. Using the array of State objects, you are to examine each state object and accumulate data.** You are to total up the state populations per region. At the end of this scan, you are to print out a header that says (see below) State Population By Region (centered) and underneath this, you are to print the region name, the number of states in the region, and the average population of states in that region. Your format for these detail lines should appear as:

```
State Population By Region
<skip line>
State          Region  Average Population per State
<skip line>
New England   6         321,123
              (or whatever average population is)

Middle Atlantic  5         x,xxx,xxx

etc.
```

(note the spacing; use the formatting links provided on my web page)

At the **end** of displaying these lines, you are to skip a couple of lines and display:

Total State Population of the Indicated Regions is: (this will be a single line)

```
State Count      nn         331,222,333
              (or whatever you add up to)
```

Note: this figure is the total populations and not the total of the averages above.  
Note also: figure out an algorithm on how to center the header above. Hint: Start with the fact that you have 80 characters of print. Use the size of the text you want to print (string length??); etc.... ☺

6. **Search Requirement. Bonus. 20 points to be applied to your Programming Grade. Please note: This will ONLY apply if everything else meets the specifications. This cannot make up for missing UML, Javadoc,**

**unmet specifications, being late, etc., okay? All else must be wonderful for this bonus to apply.**

You are to search through the entire array of State objects that are in regions 1 and 3. If any of those state populations exceed 5,000,000, you are to display: (for example)

<skip a line>

Region: New\_England

State: Massachusetts

State population: x,xxx,xxx (include the commas!)

<blank line>

Region: South

State: Florida

State capital: Tallahassee

State population: 14,345,444

<blank line>

Region: West

State: California

State capital: Sacramento

State population: 32,444,555

Etc.

You are to process region 1 before processing region 3.

Prior to displaying these detail lines, you are to display the header:

State Populations Exceeding 5,000,000 for Regions 1 and 3.

The detail lines above are to appear underneath, as displayed – a single blank line separates individual sets of three lines as shown above.

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

<blank line>

Summary Statistics:

Number of State Populations Exceeding 5,000,000: <an integer>

## **UML**

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