COP 2551 – Intro to OOP
Program #3
Due: 24 July 2007

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

Assignment #3

Objectives:

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 opportunity to create objects and develop a number of methods within their own objects
Provide student with opportunity to use StringTokenizer for tokenizing inputs
Provide student with exposure to static attributes and static methods.

Functionality:

Using States.Summer2007.txt, you are to do the following: You are provided the opportunity to read data from an input file using StringTokenizer method, but objects and some specific methods within each object, display objects to the screen as well as write them to an output sequential file, gain practical knowledge of Javadoc, as well as undertake architectural design (UML) and detail design (pseudocode). Assignment does **not** include use of arrays. (Next assignment)

1. You are to access a file names States.Summer2007.txt of unknown number of records (you may look at the file, but do **not** code your program dependent upon the number of records you will read.

2. For each line (record) read, you are to create an object of type State. You may call the objects state1, state2, etc. You will need to do file I/O to read the file. I will present examples on how to do this. You will also need StringTokenizer to read each token from each line for each object. Each time you create an object, you are to increment a static counter in the State class.

3. Create the number of objects that corresponds to the number of input records.

4. Once the objects are created, you are to access the objects you have created and display the state name, its capital and its population (use commas for the population). The display is to your computer monitor.

5. Once this is done, you are to prompt the user (me) for a state abbreviation. For simplicity, I will only submit valid state codes. You are to search each state object to see if the state code that was inputted equals the state code for 'that' particular state. When a

match occurs, you are to display the message: Match found: Input: <my input> followed by the state abbreviation, the state name, its capital, and its population. At the very end, you are to skip a couple of lines and print out the number of State objects you created in the format: (left justified) Number of State objects created is: <an integer>

Bonus: 15 points: If you edit out an invalid input and allow me three chances to input a valid one. **But be certain** everything else is perfect before you take this on, if you wish. You will only get the extra credit if all else is **perfect.**

6. You must include a UML design (your architectural design) showing object dependencies.

7. You are to include your pseudocode for your program logic (each method)

8. **ALL** methods are to include Javadoc documentation and pay close attention to any parameters that might be needed. Be certain to document them.

9. Note: You will need to import java.io and java.util. BufferedReader, FileReader, and IOException are in java.io and StringTokenizer is in java.util.

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).

'Drag' your UML design file into your P3 subfolder within your COP2551 desktop folder. It will be included in the zip file to me.

Javadoc

All programming is to be accompanied by appropriate Javadoc.

Each class and each method in each class (object) must have appropriate documentation associated with it.

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.

When you generate your Javadoc, all these comments plus a great deal more will be automatically generated for you by NetBeans.

Here's how I want you to generate your Javadoc. (Please note that there are other ways within NetBeans to generate Javadoc. But they generate lesser Javadoc than the process below. Use this approach:

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!)

You are to zip all files in your P3 as expected and Send them to be via Digital Dropbox using the same naming conventions as in P0.

Grading

Source Code – 30 points **Points earned: _____**

Indentation
Internal comments
Scope terminators
Overall program structure

Program Design – 20 points **Points earned: _____**

Appropriateness of the objects and their services provided
Interface to objects
Attribute and method visibility

Javadoc – 10 points **Points earned: _____**

Appropriateness and completeness of comments

UML – 10 points **Points earned: _____**

Correctness, associations, completeness. This means that the classes you identify are correct, that associations are indicated, and that the attributes and methods are documented within the classes.

Outputs & Program Logic – 30 points **Points earned: _____**

Accuracy and Format; Are they correct
Skip lines in between displayed numbers for readability. Make the output look professional!
Include headers / descriptors as you may feel appropriate.

Program must run correctly to receive a passing grade. Do a little at a time!