COP 3538 – Data Structures with OOP

Program #1

Due: 13 Sep Midnight

Using a current version of NetBeans, you are to write a Java program using OOP principles to accommodate the following functionality as found in the Learning Objectives.

Assignment #1

Learning 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 practice with pseudo-code for detailed program design

Provide student with exercises in Javadoc and its various formats

Provide student with exercises in searching, sorting, and comparisons of key searches and sort routines.

Functionality:

Using the external file, President.docx) you are to do the following:

1. **Build Array of President Objects**. Using the external file as input, you are to build an array of objects. In addition to your named Main class – please call it Main), you are to design and develop a second class named PresidentsMDrvr (President’s Driver) and create a number of objects of type (class) Presidents - one object in the array for each record (line) from the input file. You will need to use Buffered Reader, etc. as found (examples on Brief Introduction to Input/Output slide 7)) on my web page. Do not use Scanner.

Each Presidents object will have six attributes (properties) defined individually as integers or Strings as appropriate for each object. Recommend downloading this file first into your project folder for this project. In your project folder, you will have your source program and it may thus access the file and build the array of Presidents objects without full path considerations. (Be careful: your submitted program for grading must run on MY computer. So if you have any dependencies on yours, it will not run on mine, and that is bad news. So be careful and check!

2. Display the Array of Presidents Objects. From your main() method, you are to build and access and do a lot of processing of an array of President objects. To do this, you are to first declare an object of type PresidentsMDrvr called PresidentsDrvr in main. By declaring this object PresidentsDrvr of class (type) PresidentsMDrvr in main, you will be able to call access, build, print, etc. the array of Presidents objects from the main method, which should be a high level driver only and contain little processing.

You are to declare a class called Presidents, and within this class, you are to define all the attributes needed to define a president object and all methods that might be needed to access individual attributes of the specific object. Thus there should be a toString() method in this class and, of course, a constructor used to populate objects of this class and perhaps a few others. (You will be creating an array of 44 President objects called myPresidents within PresidentsMDrvr)

Declare an array of President objects called myPresidents is to reside within class Presidents.MDrvr. When you print out the detailed Presidents objects out, please

use the toString() method to be found in the Presidents data objects you will have created and populated.

Three **classes**: main, PresidentsMDrvr, Presidents; two **objects**: PresidentsDrvr of type PresidentsMDrvr (declared in main), and myPresidents (array of 44 President objects) declared in PresidentsMDrvr.

While this is very prescriptive for program 1, you will see the merit shortly and I will not prescribe nearly as much in the future. To sum (again) : you have a Main class, a class called PresidentsMDrvr, and a class called Presidents. Within main, you create an object of type PresidentsMDrvr called PresidentsDrvr. Within PresidentsMDrvr, you declare an array of President objects of class Presidents, called myPresidents, which will be an array of 44 objects that are to be populated.

When you display the objects in Presidents, you need to skip a blank line, then display a column header, followed by a blank line, followed by single spaced (one line of attributes per object (Presidents) as output. Only number, name, and party attributes are to be displayed per country object at this time. The column header should cite these three attributes to be printed.

Output lines are to be single spaced and all aligned nicely under their respective column header. (Again, attributes are Number, Name, and Party for this display). Numeric fields are always right justified, character or string data is always left justified. See examples of student work elsewhere on my web page for examples of formatting.). If you are uncertain about any of this description, ask early. Do not ask in ten days!

**3. Sort and Display**

**After you display the array of Presidents objects, you are to sort t**he array using an ascending bubble sort on Number and redisplay the sorted array. Use a centered header (we call this Report Title) that says: **Sorted Presidents Array.** Underneath this centered line, print the column header and detail lines as above.

**4. Sequential Search and Display Results**. Given the input search file, Presidents.Search.txt, you are to open search your sorted array of objects (object by object) to ‘find’ or ‘not-find’ each *search* *argument* in this input file. This searching of objects is to be a sequential search. Your *search key* is Party. You will need to Open and Read records from this file as above.

After skipping two blank lines, display an output header (This is merely a single line life a report header), center justification again, with the Report Title, “**Results of Sequential Search on Party**.” Then display the search argument and the result of the search as indicated in the next paragraph.

For each search argument, you are to display the search argument followed by (please space over so it looks nice!!!) the word Not Found or Found.

If Found is to be displayed, then you must to the right of this text display the number of records that match the search argument. For example,

Search Argument Result Number of Hits

Whig Found 4 occurrences

Dummy Party Not Found

Democrat Found 18 occurrences (have no idea of number)

DummyCrat Not Found

Get it?

**5. Binary Search.** This is a search again, but using a binary search with *search arguments* to *find* or *not find*.

First, sort the array of Presidents objects again on Name, ascending. For a binary search, the hit (match) must be exact. Now, for ease, I will put the search arguments right here: search on: Andrew Jackson, Abraham Lincoln, Billy Bob, George Patton, Harry Truman, XXX. (Please note: I advise you to put these strings (you do not have to do this, if you don’t want to) into an array of strings. Clearly, you ‘know’ the ones you should get a hit on and the ones you should not. But if you mistype one character, you will not get a hit, or if your ‘case’ is off, you will not get a hit. Why so??? So, these six strings are your ‘*search* *arguments*.’

You are to write a loop and successively search for these in turn. You are to format the output the same as above, with one exception: Observe:

Search Argument Found or Not Found Number of Probes

Andrew Jackson Found 4 (I have no idea how many)

XXX Not Found 6 (is 6 correct: Yes. Why?)

<whatever…>

OK

A probe is really just an ‘if.’ Where you check the search arguemtn against an element in the table. Checking a search argument against a table argument.

So the number of “probes” is the number of checks, like “is this a hit”

This sounds difficult, but it really isn’t. you will be amazed at the power you have at your disposal!!

**All done Be proud of your work.**

UML

You are to include a UML class diagram. You may use Word or Power Point. Do not use any other applications. Once you have developed your

UML (architectural design) on your desktop, be certain to *drag* this file into your project folder and be certain it is *included* with the zip file you will turn in to me. Your project folder should be named **project1.yourname**. Many students lose points here by not ensuring that their UML file is included in the project folder turned in. So please **check** the zip file before you submit. You don’t want to lose points for work done. (For this program, you should have three classes. For others, you may more, but I’m trying to show you the merit of these three classes first. We will discuss in class.

Methods in UML must be shown with visibility indicators, and number /type of arguments plus the return type. Connect all classes (label the associations). See lecture 3 (this class) for sample UML drawings. There are examples of acceptable student work on my web page as further examples. I will put these in Course Documents in Blackboard. Look up ‘Associations.’ Check out my examples on Blackboard.

One more thing: Netbeans will generate UML for you. Do NOT use it, as it does not conform to the standards I desire. I will explain in class.

Do not wait and do this at the last moment. You may be confused and lose significant points. Do this early!!

Always start with the **high-level architectural view of your solution**. You can change it easily, but it does give a great initial structure to your detailed design, that is, where the detailed processing algorithms are found.

**Detail Design**

You are to provide pseudo-code for your main (this is not main.java) processing methods. Here, it is all the ‘stuff’ in Presidents.MDrvr class. This class will likely contain code for your array builds, searches, and sorts. You do not have to provide pseudo-code for everything, but here should be your main algorithms. Examples are provided on my web page and will be found in Course Documents in Blackboard.

Javadoc – Internal Documentation

All Java programming is to be accompanied by appropriate Javadoc. Here’s the documentation I wish: EVERY method needs to have a preceding set of comments (see examples) that include three things: 1. A very cryptic statement as to function, 2. Returns: What is the return? If nothing, say nothing; if an integer, then so state; 3. Params: What are the input parameters? If none, so state; if the method is expecting something, like a name string, then so state.

Why: The documentation is a critical part of development and is for maintenance. We write programs so we can maintain them and generate revenue for the business. Basic documentation is essential for the maintenance programmer, not necessarily for the developer.

The entire deliverable is zipped via Blackboard Assignment. Be sure you have your source code, input files, UML, and detail design (in Word). It MUST be on time and not late. Programs are NOT accepted late. You have plenty of time if you start right away.

Have a folder, COP3538 on your desktop and a subfolder, P1 within it. Then P2, etc.

Lastly, be certain to look at the **grading rubric** to see how you will be graded.