Project 3

Computer Science 2334
Spring 2017

This project is individual work. Each student must complete this assignment independently.

User Request:

"Create a sortable and searchable data system for news stories and those the stories are about, that uses text and binary input and output and a graphical data display."

Milestones:

1. Implement **Serializable** for the classes necessary to save and load all application data. *5 points*
2. Use object serialization to save and load the application data to and from a binary file. *10 points*
3. Implement a simple graphical display for showing aspects of the application data. *25 points*
4. Create and/or modify classes to store information on three types of news stories. *10 points*
5. Use appropriate classes from the Java Collections Framework to save to and retrieve information on news stories and news makers. *20 points*

- ► Develop and use a proper design. (See, in particular, Milestone 4, above.) *15 points*
- ► Use proper documentation and formatting. *15 points*

Description:

An important skill in software design is extending the work you have done previously. For this project you will rework Nooz 2.0 from Project 2, adding one or more classes for additional types of data, implementing object serialization for input and output, using classes from the Java Collections Framework, and adding a graphical display. This will result in *Nooz 3.0*.

Besides the types of data described in Project 2, Nooz 3.0 will include data of another type—online news ~~sources~~ stories. Online news ~~sources~~ stories are like other news ~~sources~~ stories in most respects. However, since online stories can be posted and updated at any time, rather than being published once per day or being broadcast at a particular time, the data used by Nooz 3.0 for online ~~sources~~ stories will be based on "snapshots" of various news websites, taken at certain times during the day. In particular, each news website will have one snapshot taken in the morning and one in the afternoon.

In Nooz 3.0, we will also refine our data for TV news stories to include the notion of broadcast time, broken into four possible categories: morning, afternoon, evening, and late night.

Likewise, we will also refine our data for all news stories to include the idea of subject matter. While the topics considered previously can be seen as broad categories of news into which any given news story might fall, the subject matter of a story is more focused and particular. Since there are a great many particular subjects on which a story might report, there will be many subjects listed to cover the "big stories" of the year, yet many stories will also fall outside of these—their subject will be "No Big Story" (the story counterpart to "None").[1]

1 All of the data used in this semester's projects comes from the Pew Research Center (http://www.pewresearch.org/). In particular, all the data we've used thus far comes from the Pew Research Center's 2012 News Coverage Index Data Set (http://www.journalism.org/datasets/2012-news-coverage-index-data-set/).

Operation:

When Nooz 3.0 starts up, it will ask "Read text or binary data (t or b)?" This should be done using the technique described in the section in Project 2 entitled Reading Input from the Keyboard.

If the user answers "t" to this question, Nooz 3.0 will ask the user for a series of four text files to load. These files are, in order, a news source file, a news topic file, a news subject matter file, and a news story file. All these file names should be queried at the console and entered using keyboard input. As each file name is specified, Nooz 3.0 will read in the specified data file and store the data. The formats for these files are described in the section entitled Input Format. Next, Nooz 3.0 will ask, "Save data in binary format (y/n)?" If the user selects "y," Nooz 3.0 will prompt for a file name to which the data should be saved, then save the data in a file with the given name using object serialization.

If, instead of choosing to load the data from text files, the user answers "b" to read the data from a binary file, Nooz 3.0 will ask the user for the name of a single binary file to read. Asking the user for this filename should also be done by reading input from the keyboard. After this file is specified, Nooz 3.0 will read the file and store the data. Because the format of this data file will be determined by the serialization methods, its format is not described in this document. (Note that the first time a user runs Nooz 3.0, the user should select "t" because that is how the data is distributed initially. Only after the user has read the text data could they save it in binary format and retrieve it on a subsequent run.)

If any user-specified file is not found, Nooz 3.0 will ask for the name again until the file is found. If the user enters return without entering a file name, Nooz 3.0 will exit.

Once the data is loaded, Nooz 3.0 will enter a loop where it asks the user questions, similar to those asked by Nooz 2.0. As with the file name(s), answers to these questions should come from the keyboard. Depending on each answer given, Nooz 3.0 will proceed through a series of questions before displaying data to the user and starting over with the first question again.

The first question all users will be asked is "Search newspapers, TV news, and/or online news sources (n, t, and/or o)?" The second question is "Search news makers by exact or partial matches (e or p)?" Depending on the answer to the second question, the third question will either be "Newsmaker (exact)?" or "Newsmaker (partial)?" as appropriate. The fourth question that Nooz 3.0 will ask is, "Display text or graph (t or g)?" Depending on the answer to the fourth question, Nooz 3.0 will branch into one or another set of questions.

If the user selects text display, Nooz 3.0 will act similarly to Nooz 2.0, although with a few differences. The next question Nooz 3.0 will ask is, "Primary sort criterion is source, topic, length, or date/time (s, t, l, or d)?"[2] Then, Nooz 3.0 will ask, "Secondary sort criterion is … ?" Here the ellipsis will be replaced with the other three possible sort criteria (that is, not the one selected as the primary sort criterion). Next, Nooz 3.0 will ask, "Tertiary sort criterion is … ?" Here the ellipsis will be replaced with the remaining two possible sort criteria (that is, not the ones selected as the primary and secondary sort criteria). Once the user has answered these questions, data will be displayed to the console as described in the section in entitled Output Format. Once data is displayed to the user, the user will be asked "Save (y/n)?" If the user chooses "y," Nooz 3.0 will prompt the user "Write text or binary data (t or b)?" then ask for a file name to which the output should be saved. Nooz 3.0 will then save the data in a file with the given name. If the user has selected "t" for text output, the data will be saved in the same format as it was

2 Note the addition of date/time as a possible sort criterion. For this criterion, in addition to the actual date of the story, the time associated with the story will be considered, with newspaper stories considered to be published first thing in the morning, followed by morning broadcasts for TV and morning snapshots for online news sources, followed by afternoon TV broadcasts and afternoon snapshots for online news sources, followed by evening TV broadcasts, followed by late night TV broadcasts.

displayed to the user. If the user selected "b" for binary output, the data will be saved using object serialization. (Note that this is similar to the binary output that the user can select immediately after reading in the text data files. However, saving after reading in the data will save all of the data read in, whereas saving after searching will only save data on a single news maker.)

If, instead of selecting text display, the user selects to display a graph, Nooz 3.0 will ask, "Graph source, topic, or big story (s, t, or b)?" Regardless of the user's answer to that prompt, Nooz 3.0 will then ask, "Graph by length or count (l or ~~b~~c)?" Nooz 3.0 will then display a pie chart of the relevant data as described below in the section entitled "Graphical Display."

For most of the questions, Nooz 3.0 should only accept the one letter responses indicated as possibilities within parentheses for each question. For and/or questions, the user may specify multiple values. For example, the response "nt" to the question "Search newspapers, TV news, and/or online news sources (n, t, and/or o)?" would indicate that the user wishes to search both newspapers and TV news shows but not online news sources. Here, order does not matter, so "tn" would have the same meaning as "nt." For "Newsmaker?" any string is a valid response (but may not match with any news maker in the data). If the user enters no text, just a return, for news maker, the special news maker "None" will be assumed.

After the text or graphical display (and possible saving, in the case of text display), Nooz 3.0 will ask the user "Continue (y/n)?" If the user's response is "y," Nooz 3.0 will return to the first question ("Search newspapers, TV news, and/or online news sources (n, t, and/or o)?"). If the user's response to continue is "n," Nooz 3.0 will thank the user for using Nooz and exit gracefully. ("Thanks for using Nooz.")

If the user response to any question is incorrect (that is, does not allow Nooz to proceed to the next step), Nooz should present the user with a polite message stating that and asking the user to give new input ("The choice you entered is invalid. Please enter a new choice.").

Graphical Display:

Producing graphical displays of information can be very useful to users. Therefore, your program will have the ability to display pie charts to the user to display the data. When the user selects option "g" for graph, as described above, a graph will be displayed to the user, based on the data for the news maker found in their name search.

If the user has chosen "s" for source, Nooz 3.0 will display a pie chart with one slice for each distinct news source in the news maker's news story list. Similarly, if the user has chosen "t" for topic, Nooz 3.0 will display a pie chart with one slice for each distinct topic in the news maker's news story list. Likewise, if the user has chosen "b" for big story, Nooz 3.0 will display a pie chart with one slice for each distinct subject in the news maker's news story list. Regardless of the pie chart contents, the width (subtended angle) of each slice will be proportional to either the percentage of stories for that news maker for that distinct item or to the percentage of news coverage length for the stories for that news maker for that distinct item. For example, if a news maker has four stories in their news story list, three from one source and one from another source, and the user has selected to graph sources by count, the pie chart should show two slices, one for the first source that takes up 75% of the pie and one for the second source that takes up 25% of the pie. As a second example, if for the same news maker, the user had selected to graph sources by length, and the first three stories summed to the equivalent of 100 words but the one story from the second source had the equivalent of 200 words, then the pie chart would again have two slices but the first would take up 33% of the pie and the second would take up 67% of the pie.

Each slice should be labeled with the distinct item that corresponds to that region. To return to the previous example, if the first source was the New York Times and the second source was the

Washington Post, then the first slice should be labeled "New York Times" and the second slice should be labeled "Washington Post."

Each graphical display should have a title at the top clearly indicating the data it represents (e.g., "Romney, Mitt – Sources by Count" or "Romney, Mitt – Sources by Length." Other details of the graphical displays (such as colors used) are up to you.

Learning Objectives:

Java Collections Framework:

You have already dealt with lists as a basic data structure for storing and retrieving collections of data. However, other data structures may more appropriate for some types of collections. The Java Collections Framework (JCF) provides a diverse group of data structures for dealing with collections of different types. You should consider carefully the ways that data collections are used in Nooz 3.0 and decide on appropriate classes from the JCF to efficiently and effectively fulfill the requirements of this project.

Input/Output Formats:

News Story File

The text input file format for news stories is the same as the input file format from Project 1, with the exception that more source codes are possible (corresponding to online news sources) and that TV news stories have an extra column that indicates when the story was broadcast (morning, afternoon, evening, or late night) and online news sources similarly have an extra column that indicates when the story was captured.

New Code Files

The news code files will each consist of a series of lines, one for each numerical code and its corresponding string. The format of each line will be a numeral followed by a comma followed by the string in quotation marks. For example, a line from the source codes file might appear as follows:

```
1,"New York Times"
```

Similarly, a line from the topic codes file might appear as follows:

```
1,"Government Agencies/Legislatures"
```

Likewise, a line from the subject codes file might appear as follows:

```
2,"Immigration debate"
```

Text Output Formats

The output format text will be similar to what it was in Project 2, except that each story line should be appended with the subject and, for TV news and online news stories, the broadcast/publication time, also separated by semicolons and spaces. For example, a story line that might have ended with "`Campaigns/Elections/Politics`" in Nooz 2.0 might end as follows in Nooz 3.0:

```
... Campaigns/Elections/Politics; Immigration debate; Evening
```

How to Complete this Project:

Preliminary Design:

1 During the lab session and in the week following, you should work to determine the classes,

variables, and methods needed for this project and their relationship to one another. This will be your preliminary design for your software.

1.1 Be sure to look for nouns in the project description. More important nouns describing the items of interest to the "customer" should probably be incorporated into your project as classes and objects of those classes. Less important nouns should probably be incorporated as variables of the classes/objects just described.

1.2 Be sure to look for verbs in the project description. Verbs describing behaviors of the desired objects and the systems as a whole should probably be incorporated into your project as methods.

1.3 Also look for adjectives, if any, in the project description. Adjectives often describe features of objects that could be incorporated into your project as interfaces to be instantiated by your classes.

1.4 Write down these nouns, verbs, and adjectives (if any), along with their corresponding classes/variables, methods, and interfaces (if any).

1.5 Next, use UML class diagrams as tools to help you establish proper relationships between your classes, variables, methods, and interfaces (if any).

2 Once you have completed your UML design, create Java "stub code" for the classes specified in your design. Stub code is the translation of UML class diagrams into code. It will contain code elements for class names (potentially including names for abstract classes and interfaces), variable names, and method names; relationships between classes such as inheritance, composition, and aggregation as appropriate; variable types; and method signatures. Stub code does not, however, contain method bodies (except for return statements for methods that return values or object references – these should return placeholders such as `null`). Because we are using object-oriented design, in which the inner workings of one class are largely irrelevant to the classes with which it interfaces (that is, we are using encapsulation), we do not need to complete the implementation of any class until the design is complete.

3 Add comments to your stubbed code as specified in the documentation requirements posted on the class website. Run your commented stubbed code through Javadoc as described in the Lab 2 slides. This will create a set of HTML files in a directory named "docs" under your project directory.

4 At the end of the first week, you will turn in your preliminary design documents (see ***Due Dates and Notes***, below), which the TA will grade. **There will be no late work accepted for the design. Please note:** You are encouraged to work with the instructor and the TAs during their office hours during the design week to get feedback throughout the design process as needed.

Final Design and Completed Project

5 After the design due date, you will be provided with a UML design that you must follow for your completed project.

6 Make corresponding changes to your stub code, including its comments.

7 Create a new set of Javadoc files using Eclipse and inspect them to make sure they're appropriate.

8 Implement the design by coding each method.

9 Test each unit as it is implemented and fix any bugs.

10 Test the overall program and fix any bugs.

11 Submit your project (see ***Due Dates and Notes,*** below).

Due Dates and Notes:

Due dates:

Your preliminary design (list of nouns, verbs, and adjectives; UML; stub code; detailed Javadoc documentation) is due on **Monday, March 6th**. Submit the project following the steps given in the submission instructions **by 11:59pm**.

The final version of the project is due on **Monday, March 27th**. Submit the project following the steps given in the submission instructions **by 11:59pm**.

Academic honesty:

You may write your program from scratch or may start from programs for which the source code is freely available on the web or through other sources (such as friends or student organizations). If you do not start from scratch, you must give a complete and accurate accounting of where all of your code came from and indicate which parts are original or changed, and which you got from which other source. Failure to give credit where credit is due is academic fraud and will be dealt with accordingly.