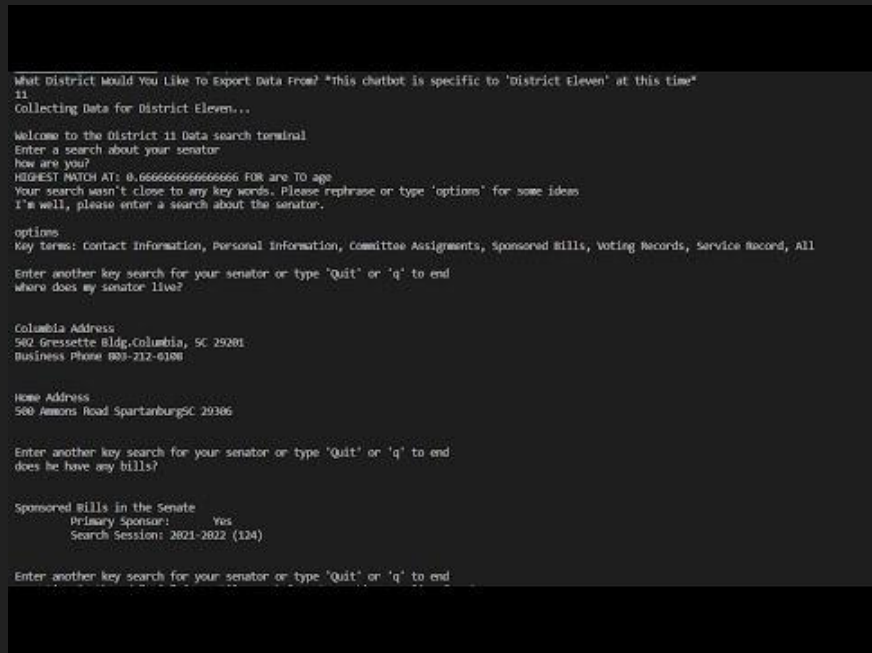
Ben Tidwell
Scope: 11, Java
Code Organization: Division of Classes

PA1: Line.replace Regex parsing
PA2: All web information is sectioned through ArrayList analyzer
PA3: Format class introduced for parsing
PA4: String Compare class with matching
PA5: Session Logger appends all data to CSV
PA6: Semaphores introduced for Session Logger

Data: All Webpage information is available. Only text results are returned (no maps or hyperlinks)

Credit of Reuse: “CommandQueries” from Ryan Capron’s github



Requirement – What did the instructor ask you to do?

To create a chatbot that addresses any user Query and returns information from a Senators District site. The bot should also log prior session and have data from sessions retrievable. Specific queries below…

[#1] "Quit" or "quit" or just "q" => Program exits
[#2]"Tell me about the representative", "Tell me about the rep" => Personal Information (Type-I2)
[#3]"Where does the rep live" => Contact Information (Type-I1): Home Address
[#4]"How do I contact my rep " => Contact Information (Type-I1)
[#5]"What committees is my repo on" => Committee Assignments (Type-I3)
[#6]"Tell me everything" => Give all information Extracted
[#7] "What district do you support for Q/A" => Give district number and name
[#8] *<User can enter any other text and the program has to handle it>* => "I do not know this information" or
"Here is my guess - " + <query> + <answer>. "Did I answer correctly ? "

myrep-chatbot –summary
=> There are 12 chats to date with user asking 23 times and system respond 24 times. Total duration is 456 seconds.
- **myrep-chatbot** –showchat-summary 2
=> Chat 2 has user asking 2 times and system respond 2 times. Total duration is 4 seconds.
- **myrep-chatbot** –showchat 2
=> Chat 2 chat is:
…
- **myrep-chatbot** –showchat 200
=> ERROR: there are only 12 chat sessions. Please choose a valid number.

Specification – What did you you do, what scope you selected and what decisions you made?

Utilized Java for analysis of the district 11 page.

Began with a formatting class. This class pulls all data from the website at the time the program begins so all information is always up to date. This formatting class parses the information to remove html spacing and syntax.

I decided to have the users search parsed to identify specific words that can address the site. If the user contains a key search word or phrase by 70% match, then that is used to address what data is returned.

Development highlights – How was your code implemented, e.g., module design, classes

Classes

- Compare: Compares similarity of string at a percentage. Used to evaluate the user input to key search phrases
- Duration: Pulls the users local time at the beginning and end of the program. Returns the difference for the length of session
- Format: Downloads most current website and parses information
- Sessionlogger : Appends to a CSV file the last sessions searches, responses, and data.

Development highlights – How did you test ?

Testing:

Testing was done with the professors listed required query search addresses. Testing was also done using typos, odd punctuation, incomplete words, non words, etc. Roommates were used to test their own searches and provided feedback

All testing was manual.

Integration testing was used mainly on the formatting class and duration class.

Development highlights - What problems did you face and how did you solve them?

Problems Faced:

Error exceptions and poor performance when checking integers of every line from the CSV for the session duration. Instead a semaphore was used when the next line was expected to be the session duration.

Issues with printStream destinations being overlapped. Instead the majority of print streams are updated sequentially all from the formatting class then the sessionlogger is called at the end of the program to update the CSV file.

Reuse – What did you do to make your code reusable?

Through divisions with classes

Compare class is very reusable, it takes parameters of two strings and returns similarity value.

Duration is very reusable. By calling an instance of the class any program can use the start and end methods sequentially to have an accurate time between returned.

Formatting is reusable through the way in which a website is called and downloaded. Parsings may need to be changed for accurate results from other districts.

Sessionlogger is reusable for the purpose of returning parsed sessions. The only parameter needed is a desired session number search and so long as there is notation for the “End of Session” any file can be addressed.

Reuse – Whose code did you use and why?

I utilized “CommandQueries” from Ryan Capron’s github. I had difficulty on VS code utilizing command line arguments and so testing the development of this section became very tedious. I utilized Ryan’s section for the sake of my own time and sanity then ultimately tested through Eclipse.

(https://github.com/hatc3000/CSCE240/blob/main/prog_assignments/prog5-sessionlogger/src/commandQueries.java)

```
if (args.length > 0) {
        if (args[0].equals ( anObject: "summary")) {
            log.allSessions();
        }
        else if(args[0].equals ( anObject: "showchat-summary")) {
            try {
                chatNum = Integer.parseInt(args[1]);
            }
            catch (NumberFormatException e) {
                System.out.println( x: "Invalid ChatNum");
            }
            log.summarySession(chatNum);
        }
        else if(args[0].equals ( anObject: "showchat")) {
            try {
                chatNum = Integer.parseInt(args[1]);
            }
            catch (NumberFormatException e) {
                System.out.println( x: "Invalid ChatNum");
            }
            log.showSession(chatNum);
        }
    }
    //End of derivative
```

Reuse – What challenges did you face? • Future work - What more can be done to make your chatbot useful?

What Challenges:

It was challenging to manipulate command line arguments through VS code. Parsing and user query handling was tedious to begin but with spread classes who address components it was easier to handle.

Future Work:

If a more sophisticated parsing method were used for the website then it could be very quickly ported to benefit all district sites.

Reuse – How will the code need to be changed over time?

The code needs to be optimized. This can be done in sections of the line parsing for the site data, sections of addressing the user string comparison to key words/phrases.

The session logger uses many variables that can be globalized rather than reinitialized. Many of the session logger methods have overlapping actions that can be classed and decentralized.