The purpose of this project was to create a chatbot for answering questions about the life of famous music artist Post Malone. The chatbot was coded in Java as it was the language that all collaborators were familiar with.

The Java code is organized into 2 classes that use 2 separate text files for both greeting the user and looking up responses to user input, named greeting.txt and responses.txt respectively.

The **Response** **class** contains a **hash map** in the form of <user input : response> and 2 methods. The **greeting()** method reads the greeting.txt file and generates a greeting message for the user upon bootup, with an error case for unreadable text files. The **loadResponses()** method reads the responses.txt file, splits them along the “:”, puts each half into a string variable and puts these strings into the ‘responses’ hash map.

The **Main** **class** extends the Response class and contains a linked list, a scanner object, and 3 methods. The **receiveInput()** method prepares the chatbot for user input using the scanner object and by displaying a message prompt to ask our chatbot something. The **parseInput()** method refines user input for our chatbot and adds this refined input to the linked list of responses. The **main()** method consolidates the 2 aforementioned methods as well as the loadResponses() method of the Response class. A response object is created, the reponses.txt file is stored inside this object, the greeting is displayed, input is received and parsed, and the resulting output is displayed to the user. There is also an error handler for responses made from no user input.

## Second Topic for Responses

A second set of responses was added to the Response dataset, allowing for more diversity in the conversation. Since our chatbot is essentially a simulation of a conversation with Post Malone, adding another topic surrounding music artist, adds depth to the conversation, where one can ask Post Malone his personal opinion on artists.

## Synonyms

The synonym recognition system uses Stanford NLP’s WordNet database and MIT’s Java WordNet Interface (JWI) to lookup synonyms from the responsesTagged.txt file. It consists of 2 classes namely WordNet.java and DemoWordNet.java.

WordNet.java contains 2 methods namely loadResponsesTagged() and searchWord(). The first method reads the responsesTagged.txt, splits each line of the file by “/” and creates the keyword : response format similar to the loadRespnoses() method. The difference in this method is that the keyword and the POS tag are separated so that when the tags are passed into the searchWord() method, the chatbot can properly lookup synonyms from the WordNet dictionary object. The searchWord() method creates an indexed word object with the value of the tagword from loadResponsesTagged() and looks this word up in the dictionary, gathers the synonyms of the word and returns the glossary of synonymous words.

The DemoWordNet class extends the WordNet class. It creates a WordNet object, calls the loadResponsesTagged() method, and uses a for loop to iterate over every key in loadResponsestagged.txt, gathering the synonyms.

## Off-Topic Responses

An Array data structure was added and used to store 5 off-topic responses. The index into the array is generated by the **Math.Random()** function. Once the index is generated, it chooses the response from the array at that index and displays it to the user. This helps continue the conversation in the event that the user types something that is not within the set of key words we previously defined in the other topics.

## POS Tagging

The POS tagging system, looks at our current responses.txt, and then assigns all keys a POS tag. The tagging class contains three functions, translateResponses(), findKey(), and callTagging(). The Translate responses function looks at all the current responses that were already in responses.txt, and creates a tag for each key in the file. The callTagging function is called from main, and it looks at user input and assigns tags to each input string. The findKey function is called from main it finds keys associated with values in order to find more responses.

The POS tagging system improves the abilities of our responses in a slight manner, the improvement stems from the fact we can look through input sentences and see if the noun or verb is being used in the same manner as in the key response pairs in our file. If the pairing is not the same it prevents false matches as words that can be used both nouns and verbs are no longer applicable answers.

## 

## GUI

The GUI was built using the Swing java toolkit. Although this toolkit may be a little outdated, it is effectively good for learning the basics of GUI development. Upon running the application, the user is presented the GUI window where they can then interact with the Post Malone bot. Adding a GUI makes the program feel more professional and is therefore more inviting for the user to engage with it further. The GUI acts as the link between the user and the bot while keeping a brief log of the recent chat to see the conversations history. Small visual cues were added to provide feedback to the user such as

## Notes

In order to run the project one must navigate to buildpath -> configure buildpath, select libraries and include the edu.mit.jwi\_jdk.jar, and stanford-postagger.jar in the class path. Additionally, you must ensure you are running JAVA SE-13 or better.

**Students**

**Samson Kneller**

**Dalmarr Hussien**

**Nick Hemingway**

**Carson Perreaux**

**Tanvir Kang**