COT6930 NLP Justin Johnson

Dr Dingding Wang Z23136514

**Assignment 1**

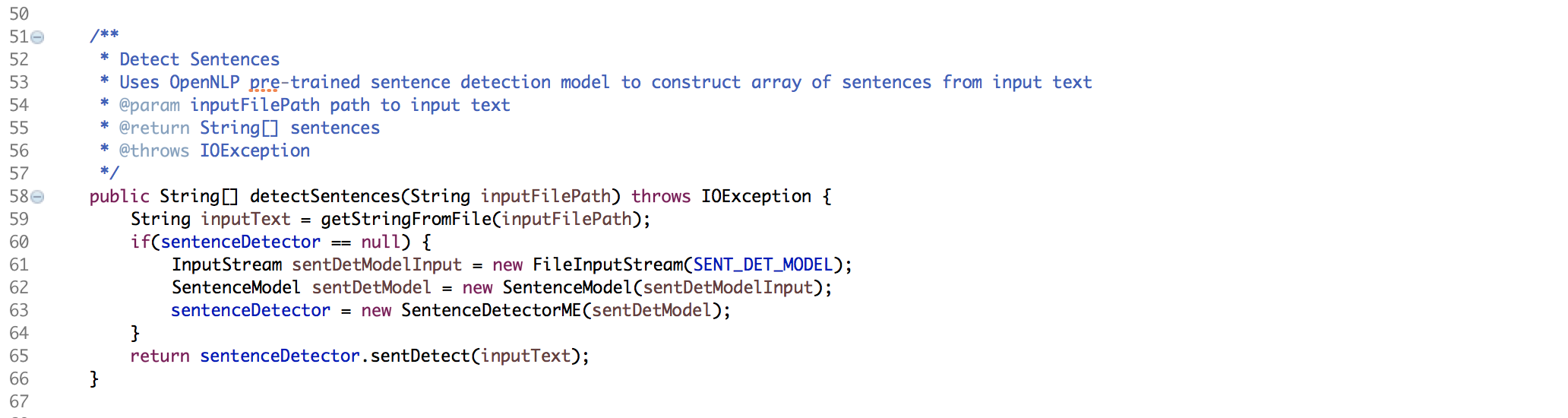
**Part-of-Speech with Apache OpenNLP**

**Introduction)**

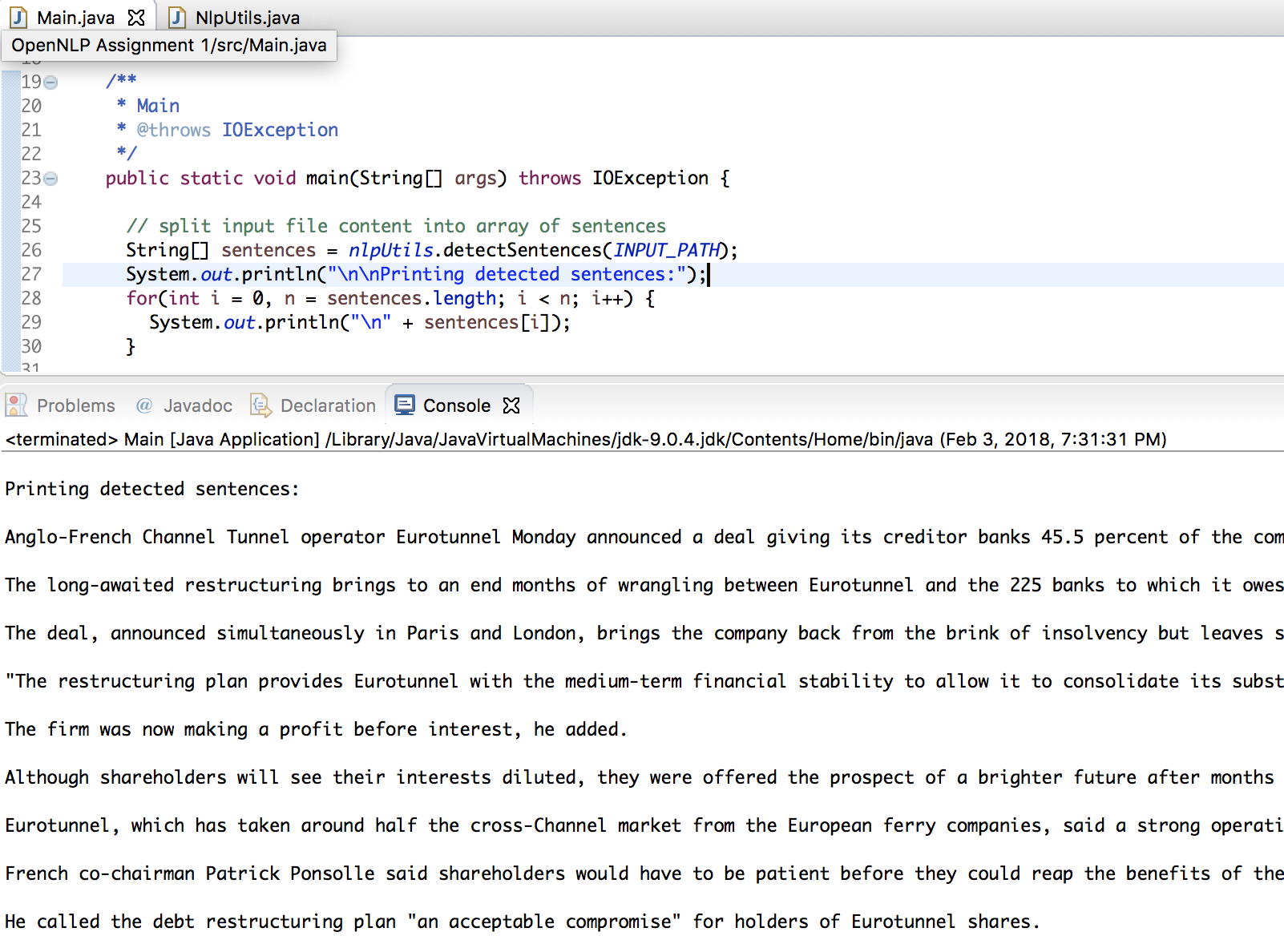
Sentence detection, tokenization, part-of-speech tagging, and name entity detection are performed on a one-page news article using Apache’s OpenNLP library. OpenNLP is a machine learning toolkit for Java. OpenNLP also provides language specific pre-trained models that are used to complete several steps of this assignment. Each step is explained with screenshots of function definitions and results.

**Part I) Sentence Detection**

Sentence detection is achieved by reading a input file into a String, then passing the input String to an instance of OpenNLP’s SentenceDetectorME class. The SentenceDetectorME object is instantiated with OpenNLP’s pre-trained model, en-sent.bin.



Below is a screenshot of the above function being invoked, and the corresponding results being printed to the console. The result is an array of sentences, as expected.



**Part II) Sentence Tokenization**

Next, each sentence is tokenized using OpenNLP’s SimpleTokenizer. Provided a string, the tokenizeString function returns an array of tokens. The tokenization results will be displayed in Part III, with the POS results.

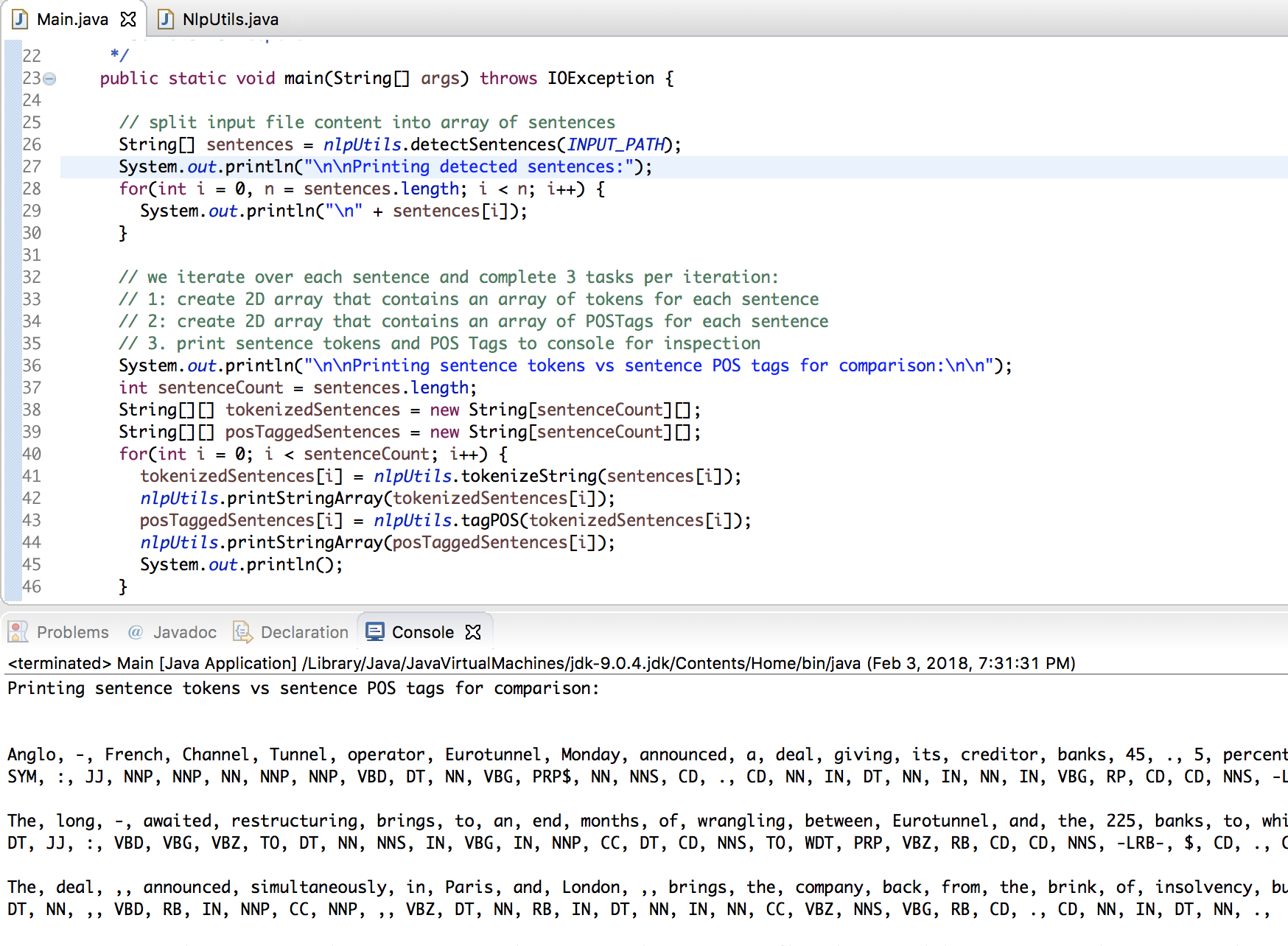


**Part III) Part-of-Speech (POS) Tagging**

Part-of-Speech tagging is completed using OpenNLP’s POSTaggerME object. A POSTaggerME instance is initialized with OpenNLP’s pre-trained model, en-pos-perceptron.bin. Given an array of tokens, the POSTagger returns an array of POS tags, such that the resulting array’s ith tag corresponds to the input token’s ith token.

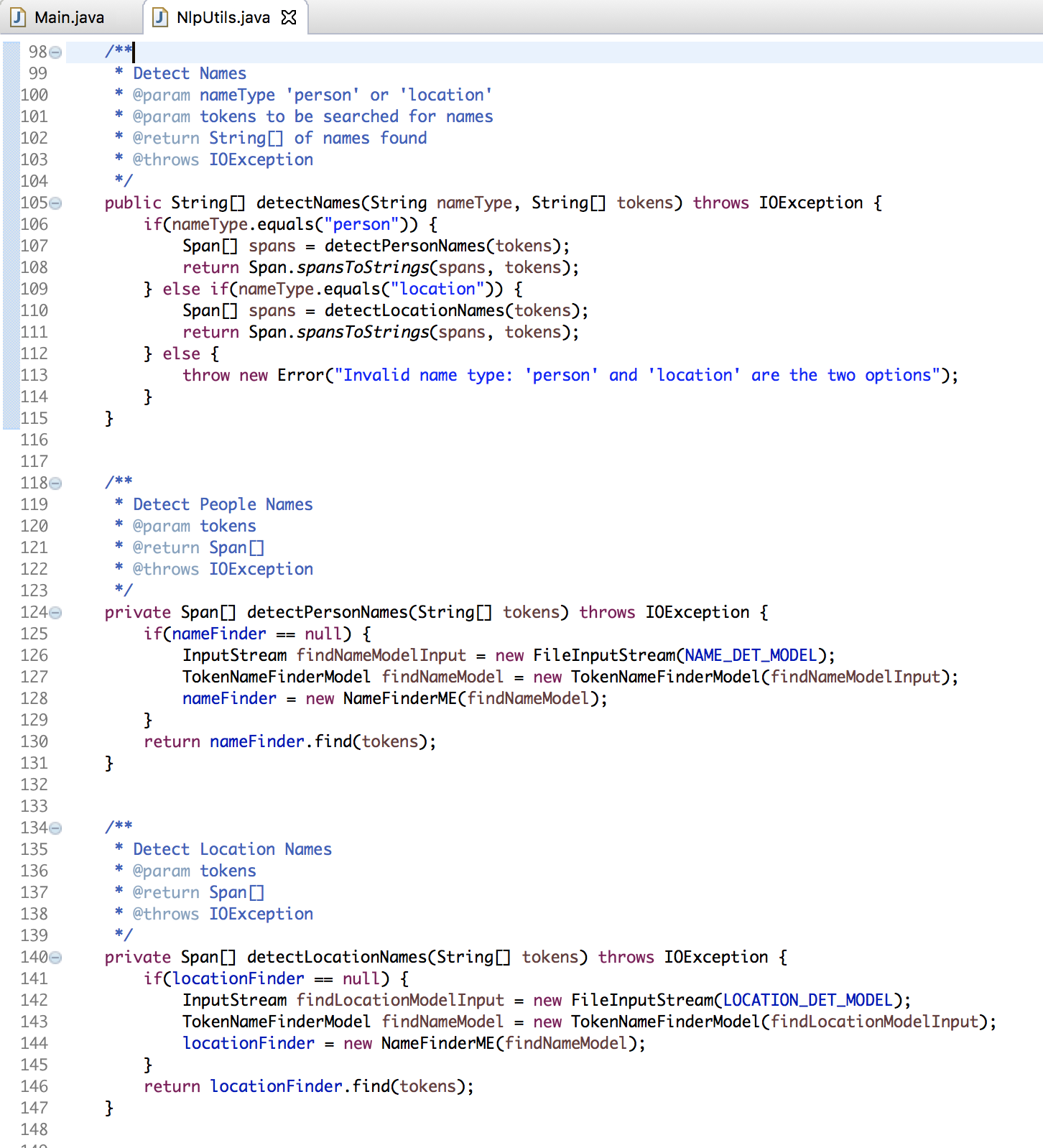


The following screenshot displays tokenization and POS tagging in use, followed by the results. The array of sentences detected in Part I are traversed, creating an array of tokens for each sentence. The array of tokens is then passed to the POS tagging function, which then returns the POS tags that correspond to the array of tokens. Once completed for each sentence, the sentence’s tokens are printed, and then the sentence’s POS tags are printed.

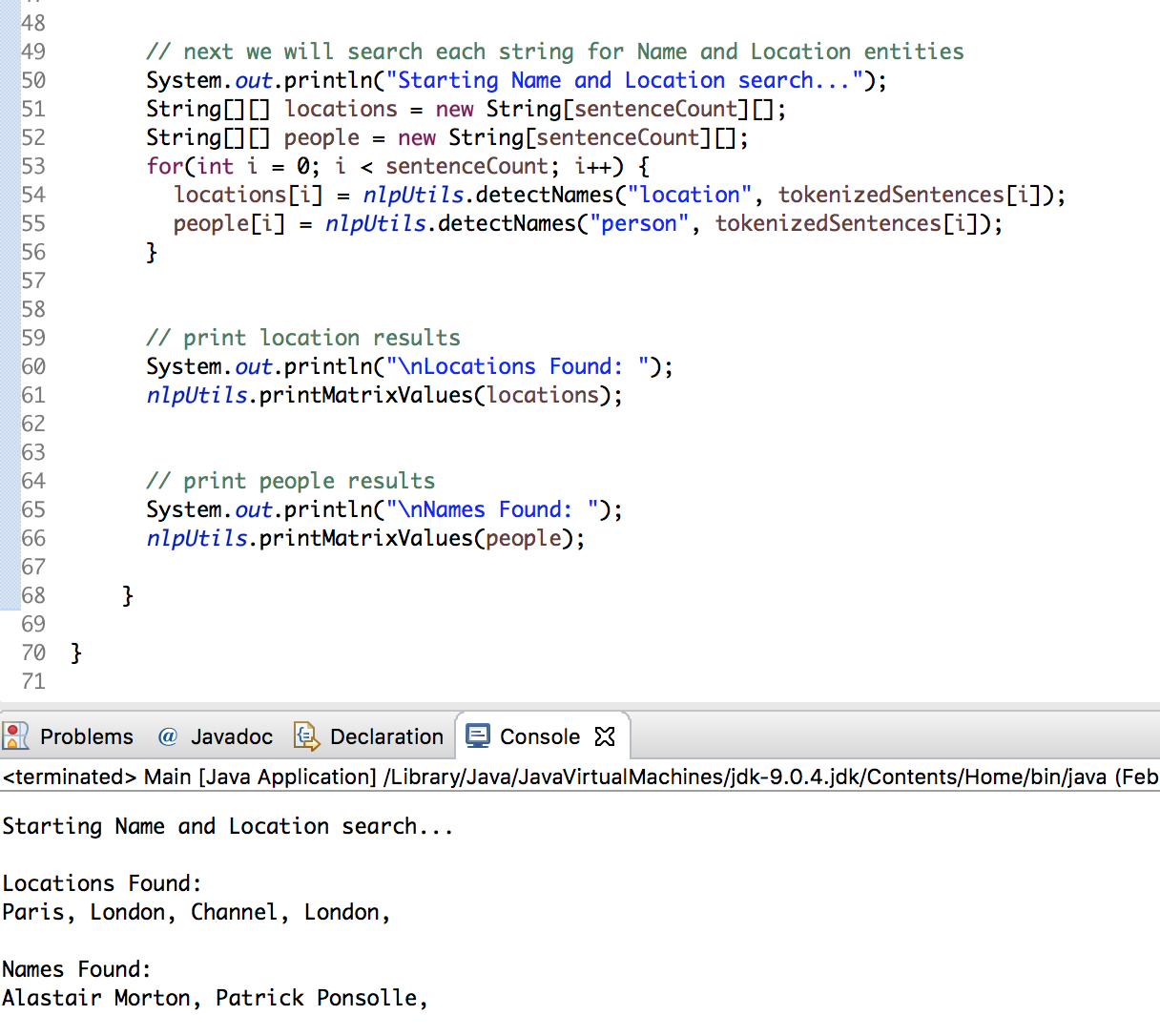


**Part IV) Name Entity Detection**

Finally, all sentences are checked for Name Entities (person and location). As displayed in below screenshot, OpenNLP’s NameFinderME class is able to detect names for both people and locations. One instance of NameFinderME is given OpenNLP’s pre-trained person entity model (en-ner-perons.bin) and the other is given a location entity model (en-ner-location.bin).



The models were able to identify 4 locations and 2 people. One word, Channel, was improperly identified as a location.



**Conclusion)**

OpenNLP’s natural language toolkit was successfully used to perform sentence detected, tokenization, part-of-speech tagging, and entity detection. Sentence detection, POS tagging, and entity detection utilized OpenNLP’s pre-trained models. Tokenization was completed using OpenNLP’s SimpleTokenizer.

The news-article.txt input file was then examined to evaluate the results of person and location entity detection. The name entity detection correctly returned 2 of 2 names from the news article, achieving 100% accuracy and 100% recall. The location entity detection selected 4 locations from the article, 1 of which (Channel) is a false positive. Location entity detection therefore obtained 75% precision and 100% recall.