**COSC 320 – 001**

**Analysis of Algorithms**

**2022/2023 Winter Term 2**

**Project Topic Number: 50**

**Keyword replacement in a corpus**

**Milestone 4**

**Group Lead: Zhiheng Zhang**

**Group Members:**

**Zhiheng Zhang**

**Xinquan Ge**

**Analysis of the first algorithm:**

The first algorithm splits the text by reading the input text byte by byte and saves all the letters together as a word when it reads the next place is a space/punctuation. The words read are stored in a content storing array list, containing the word being read and the punctuation by sequence. Then the algorithm reads the array list that stores the abbreviation and the whole word, if the word being taken out equals to the abbreviation column in the replacing array list, then replace the word with the whole word column in the replacing array list, otherwise the word does not need any operation and directly store it in the array list. Keep reading and replace each word being read until all text is finished reading.

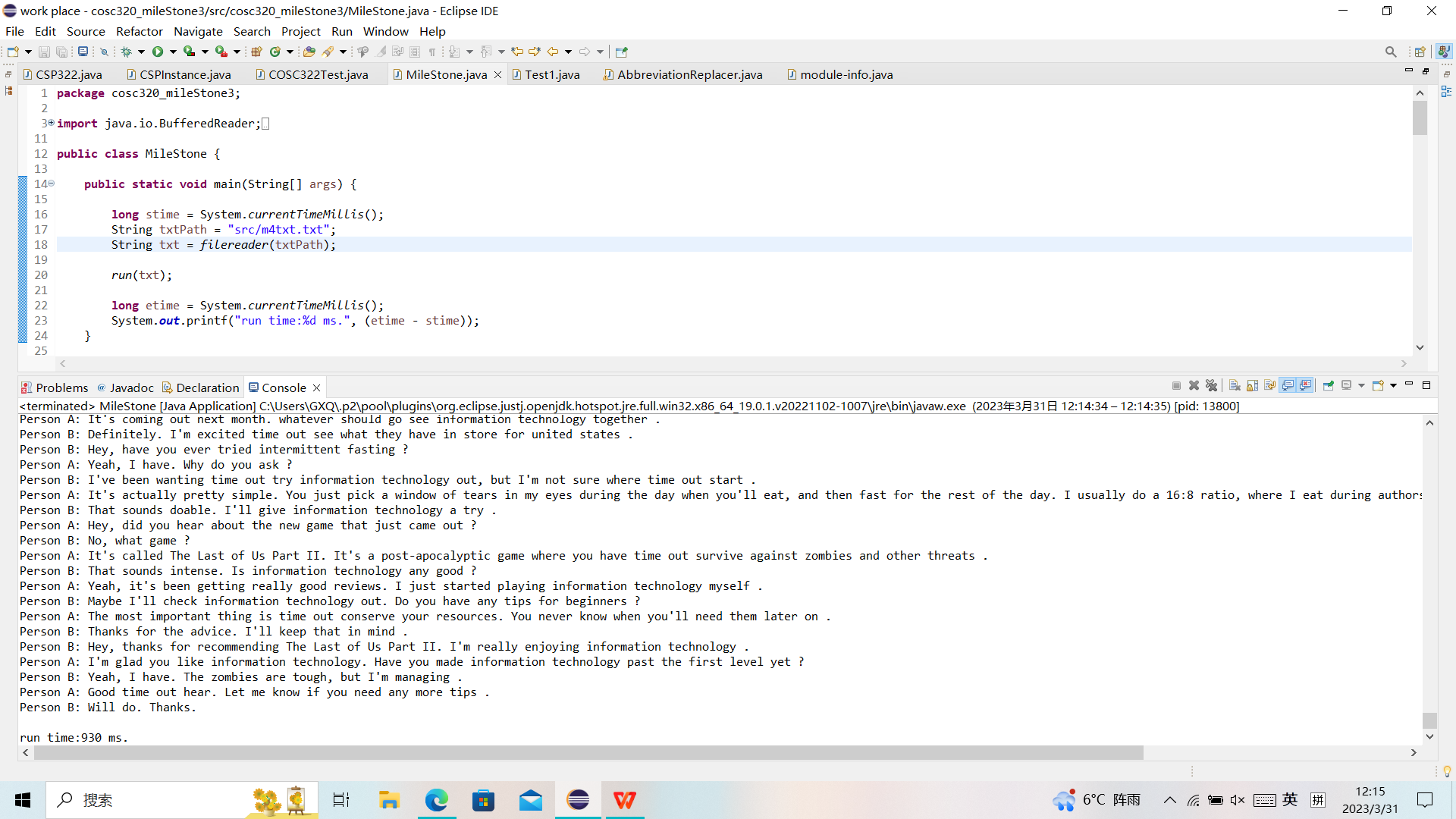
After all text is finished with reading and replacing, merge everything in the content storing array list now, it will give the original text with all the abbreviation being replaced with whole words. The most time consuming part in this algorithm is the searching and replacing part, especially sometimes there might be abbreviations containing lower / upper case which is different as those stored in the replacing list but being the same abbreviation. The algorithm force the word being taken out is changed into lower case so it equals to those in replacing list, while this increases running time a lot.

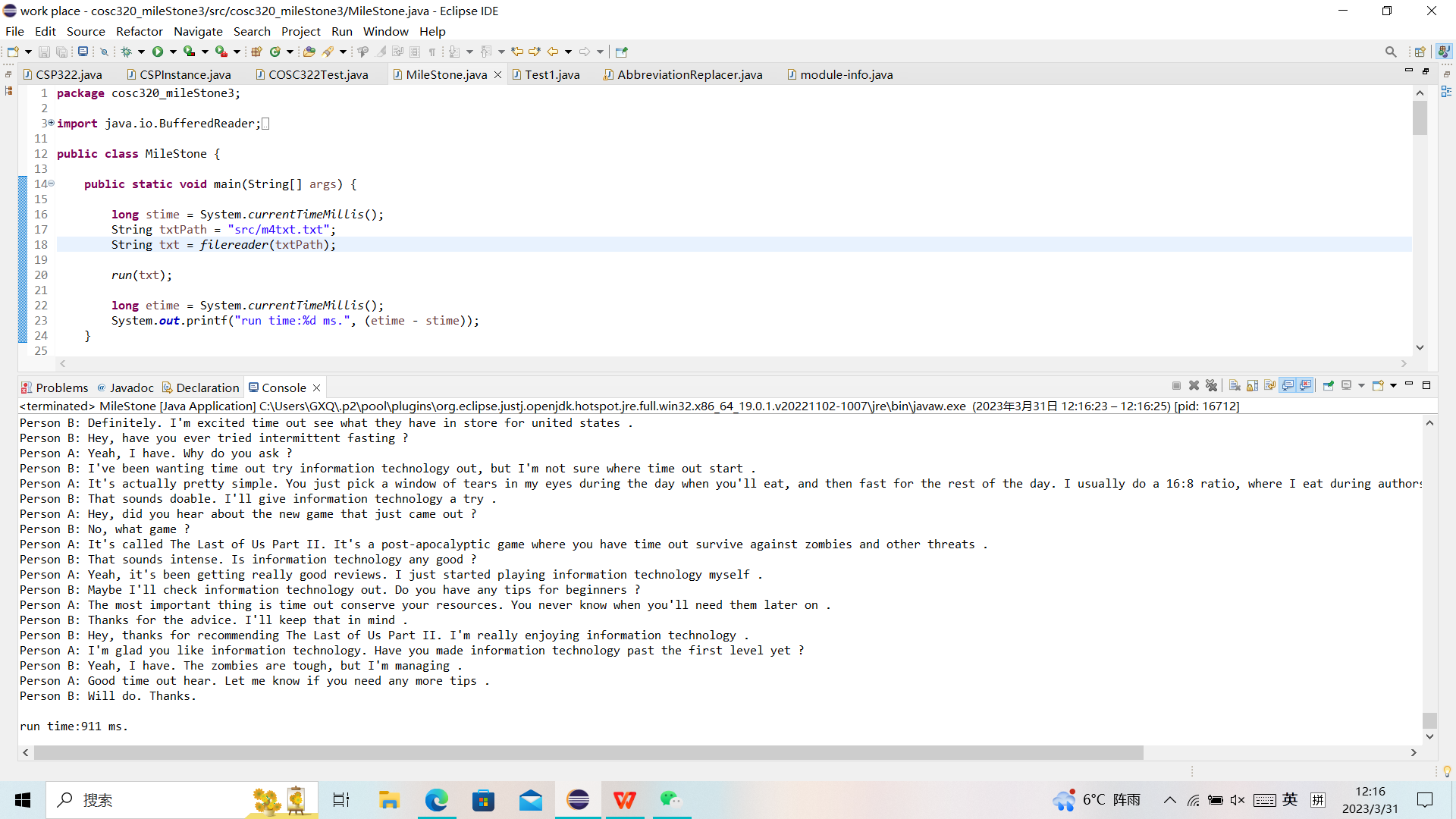
**Analysis of the second algorithm:**

This algorithm is the same as the first one when doing splitting text content and merging the content back together. The different part of this algorithm with the first one is the data structure we choose to store the replacing list. This algorithm uses Hash Map to store the replacing list.

When the algorithm is searching an item in an array list, it will check if the array index is over the limitation, if not, it reads the content by that index and return what it gets. By contrast, Hash Map does not need to check index overflow and it directly reads the content rather than using index. This will save much time when the search set is large and makes the second algorithm run time shorter than the first one. However, the usage of Hash Map actually makes the algorithm using more space to store the data, since Hash Map is a data structure that is unordered and array list is ordered. The running time being saved is actually like a deal by paying more storing space.

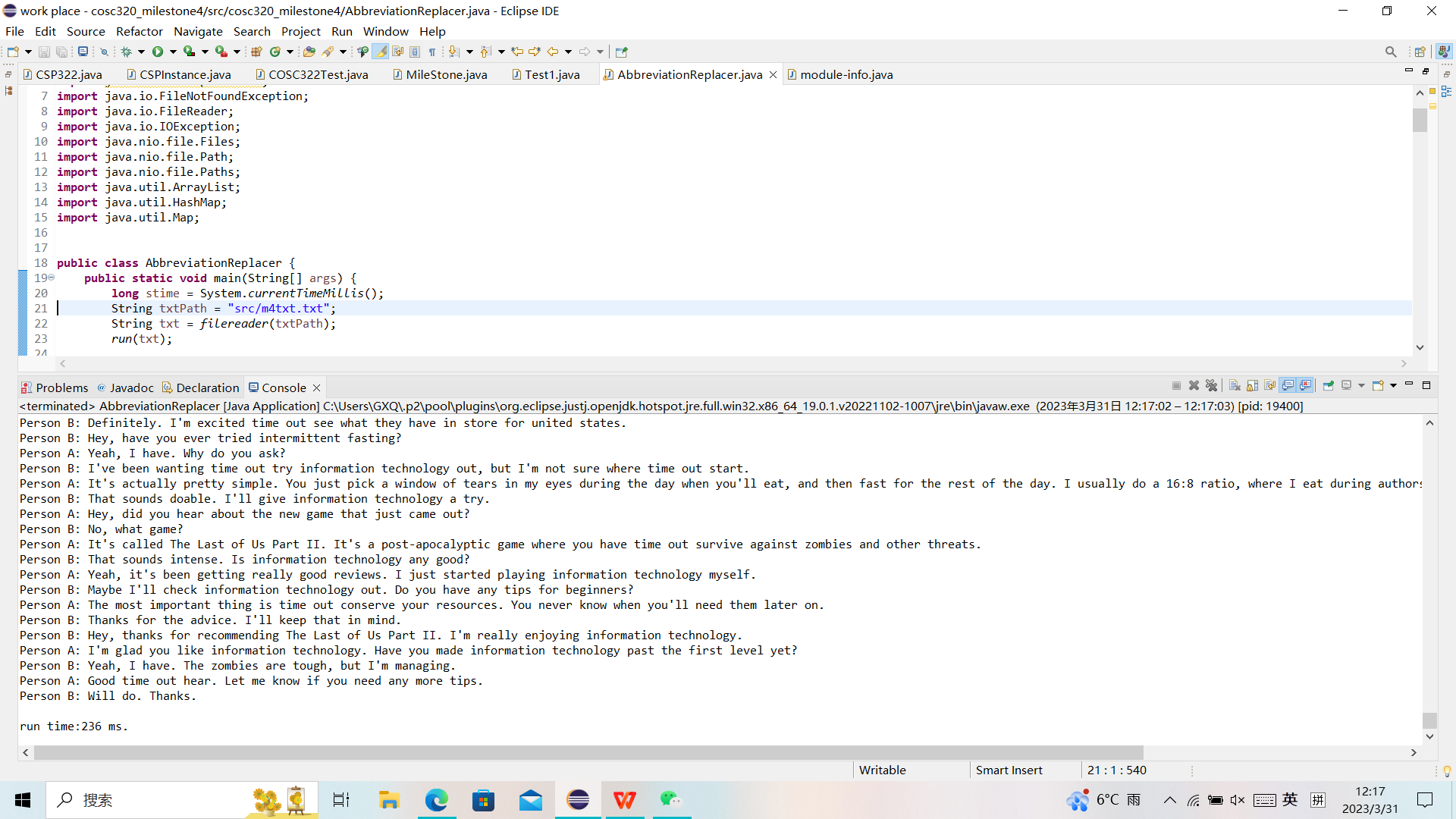
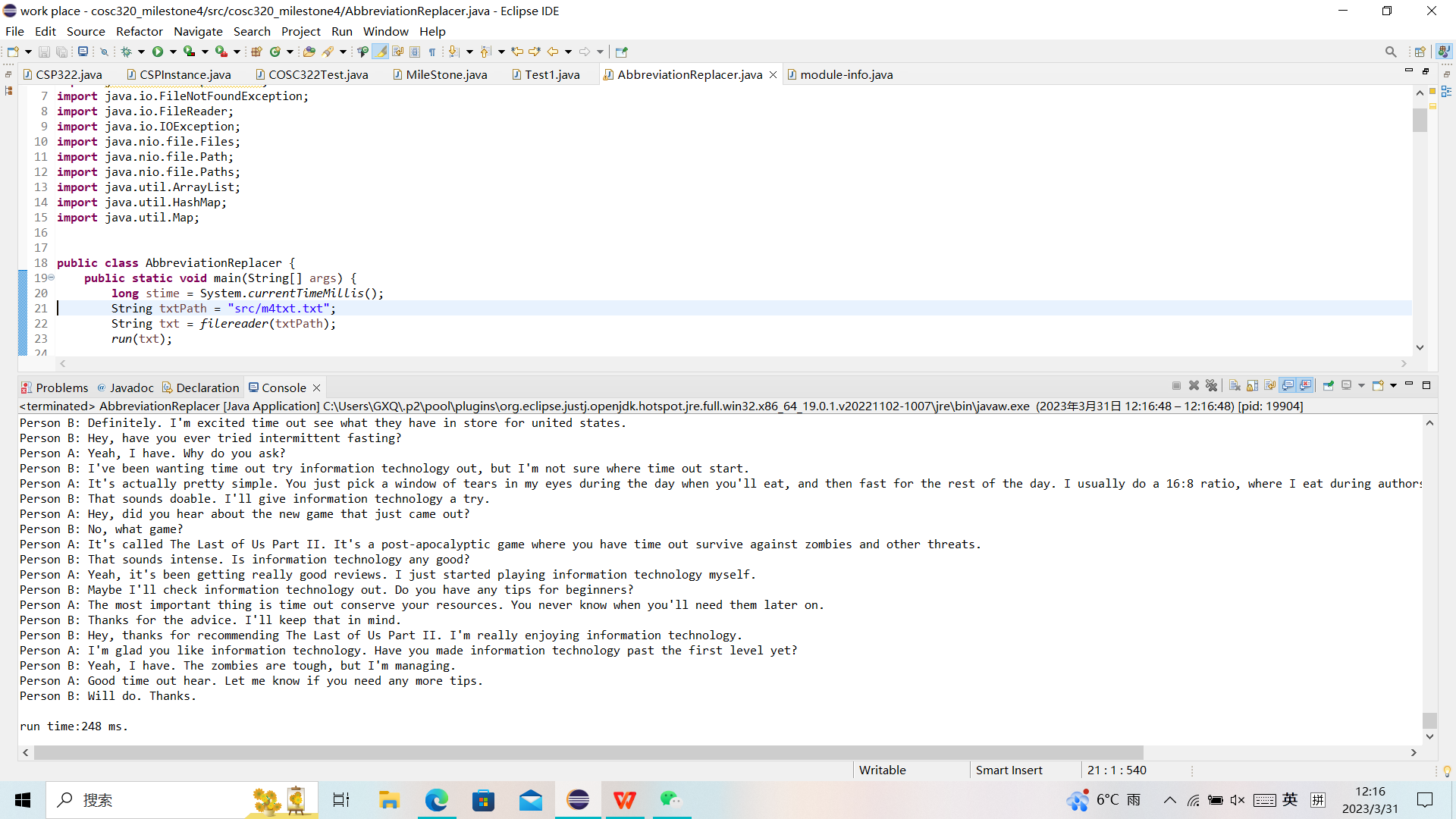
**Running Outcome of First Algorithm:**

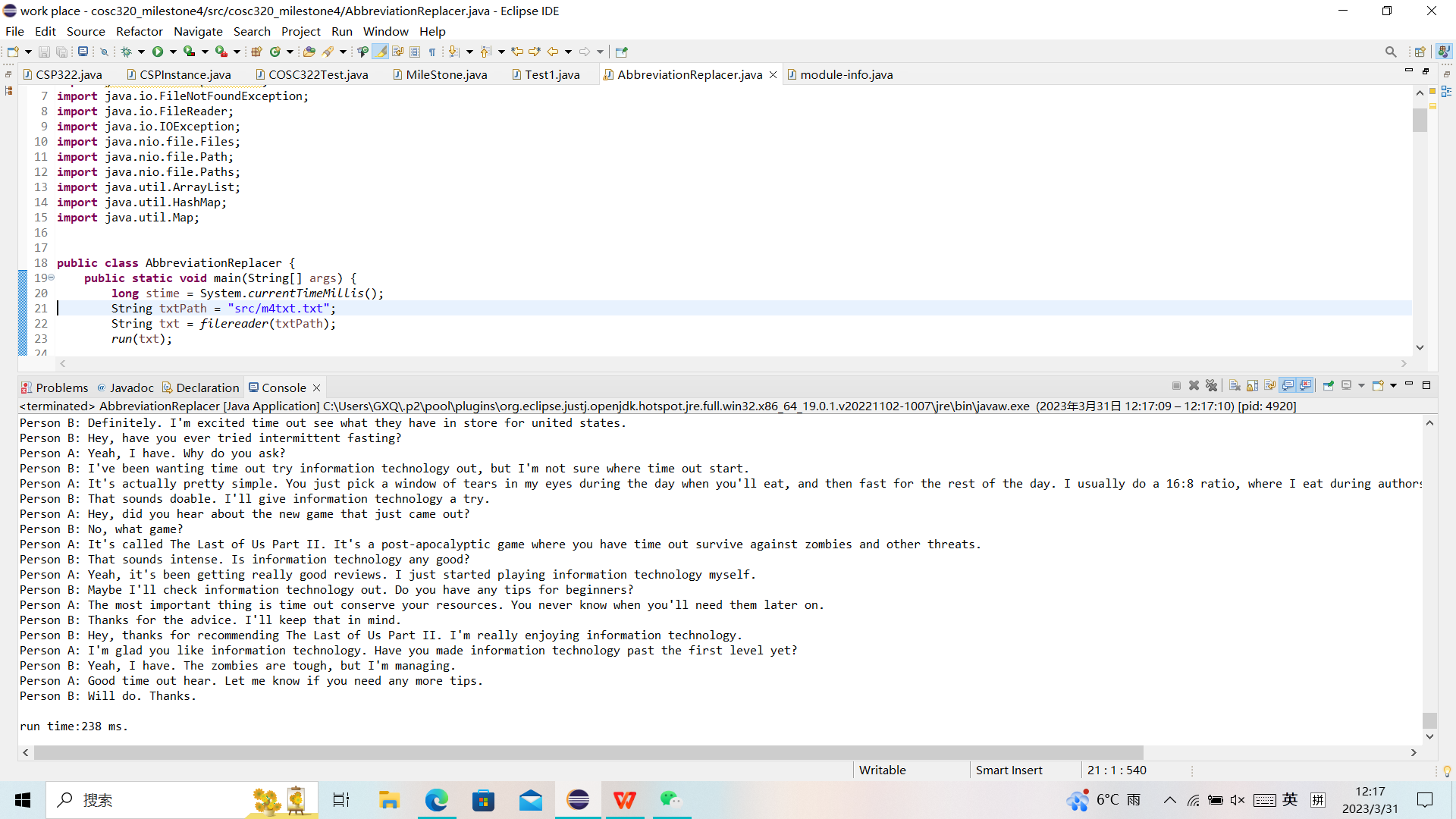




Run time: 940ms 911ms

**Running Outcome of Second Algorithm:**





Running Time: 248ms 236ms 238ms

**Conclusion by comparing runtime of two algorithm:**

The runtime of the second algorithm is obviously much shorter than the first algorithm, which is the same as we expected. There is no cases that the second algorithm need more time than the first one, which means the observation of faster run time is not a coincidence. Hence, we can conclude that the second algorithm is a better algorithm than the first one in terms of running time.

GitHub Web Link：

https://github.com/ZhiHengZhang01/COSC320-Term\_project

**Work Division:**

Zhiheng Zhang:

first dumb version of algorithm

Programme debug

report writing

Video recording and editing

Xinquan Ge:

continue on the first dumb version

Programme debug