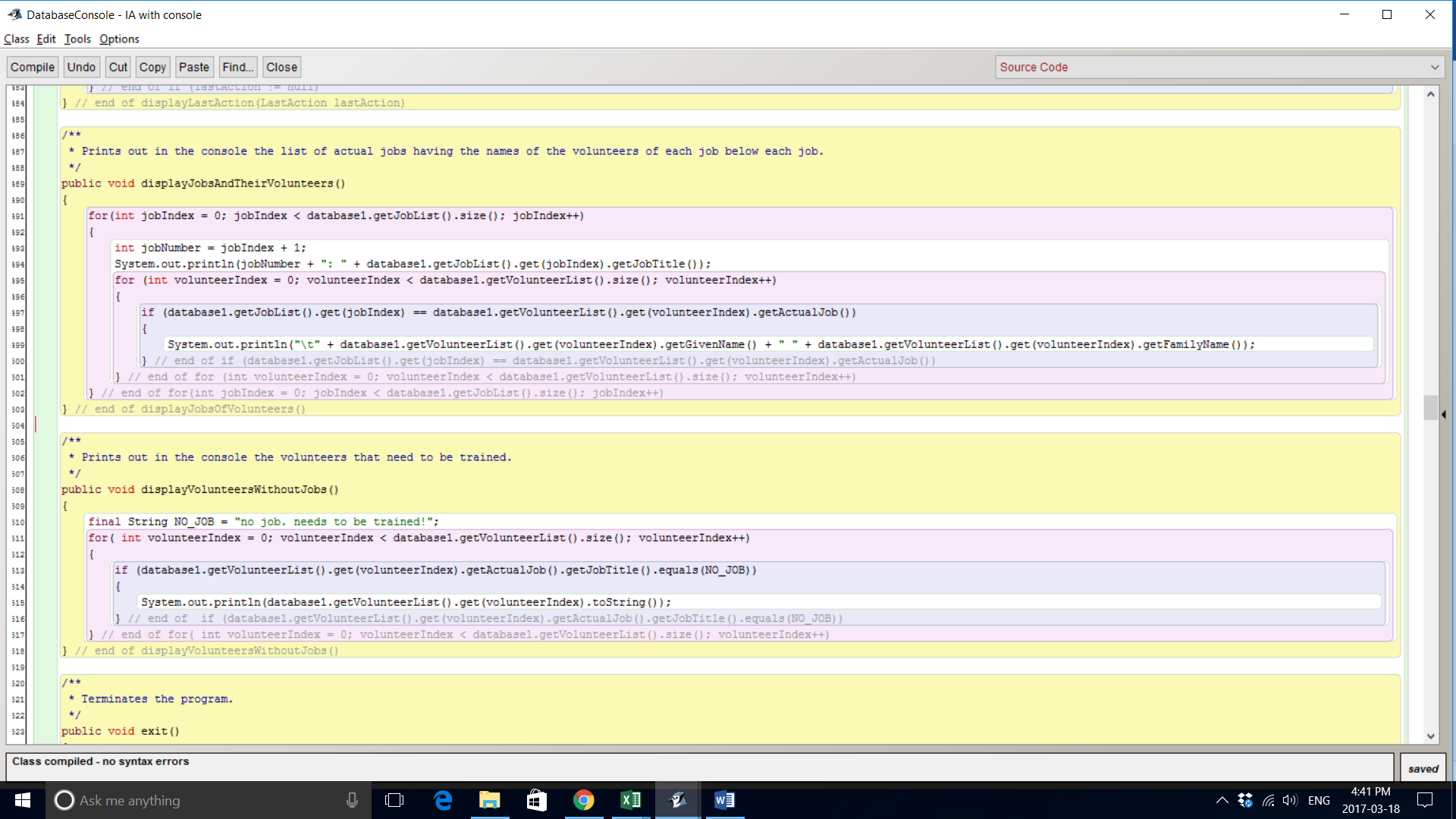
Criterion C: Product Development

# Techniques used to create the database

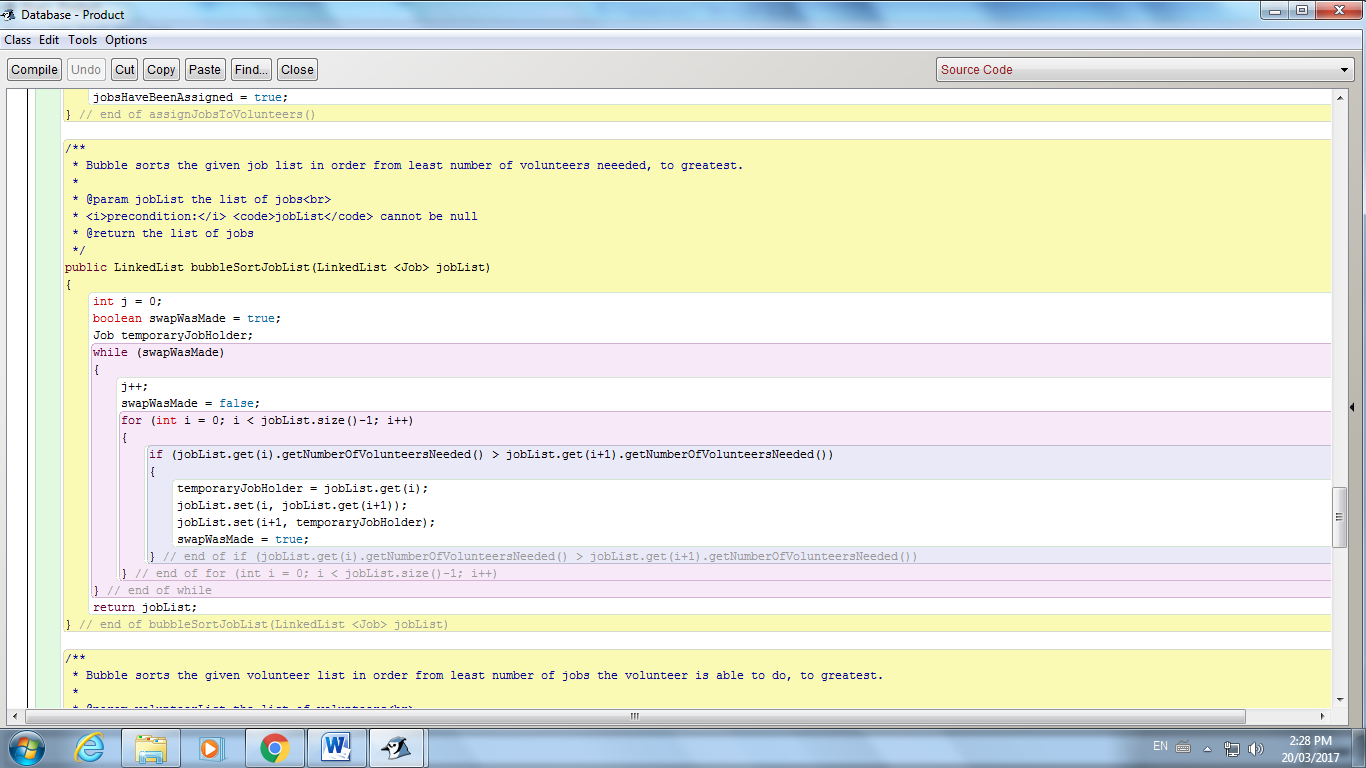
* Searching and Sorting
* Algorithmic Thinking
* Java Console Input/Output
* Dynamic Lists
* Error Control
* File I/O

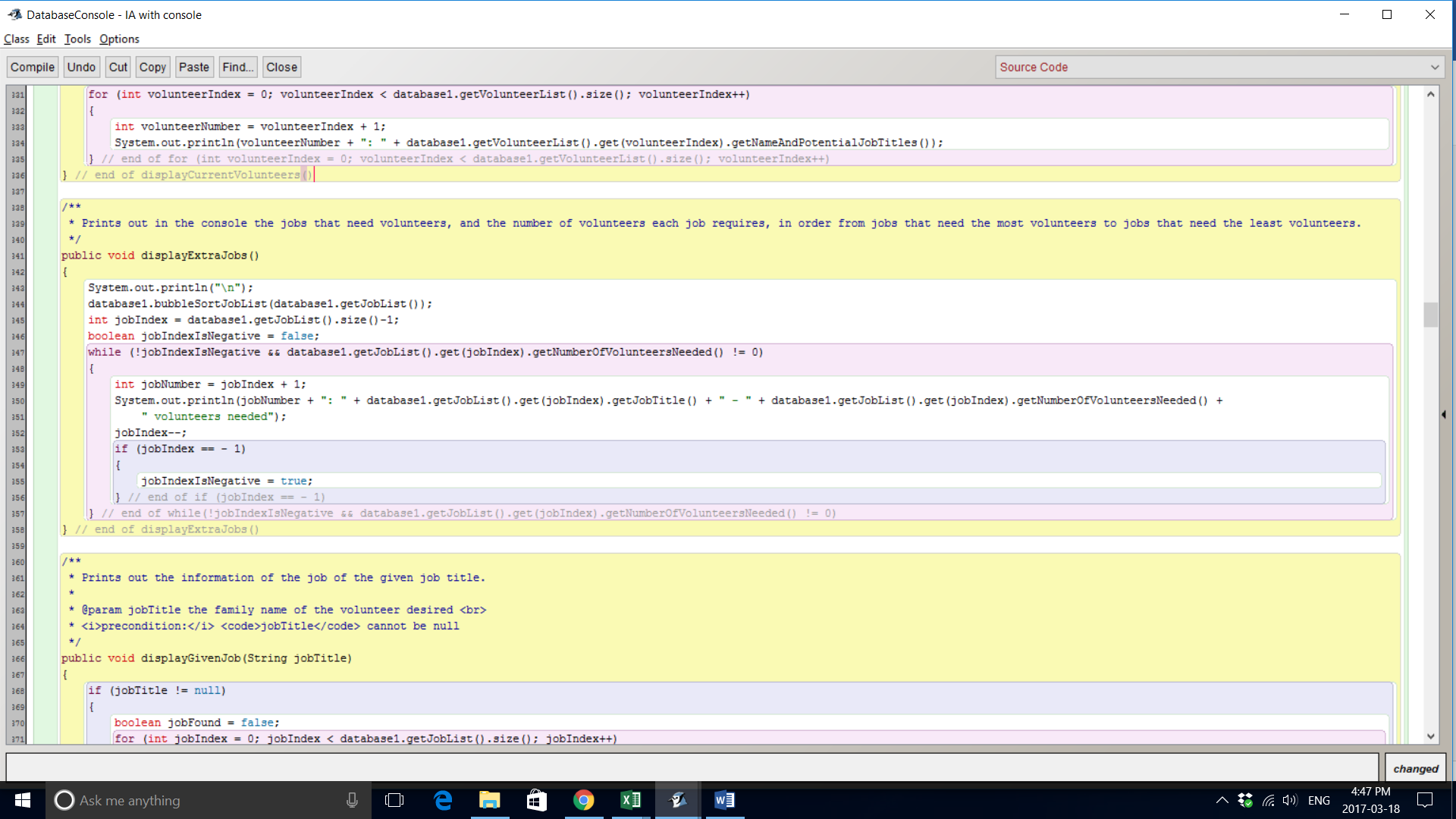
## Searching and Sorting

The relational database uses both searching and sorting algorithms to maximize its efficiency when doing tasks.



The method above uses a sequential search to linearly search through the volunteer list to print out all the volunteers without jobs. A sequential search is necessary as there is no way of sorting between if a volunteer has or does not have a job (i.e. Boolean instance variables).

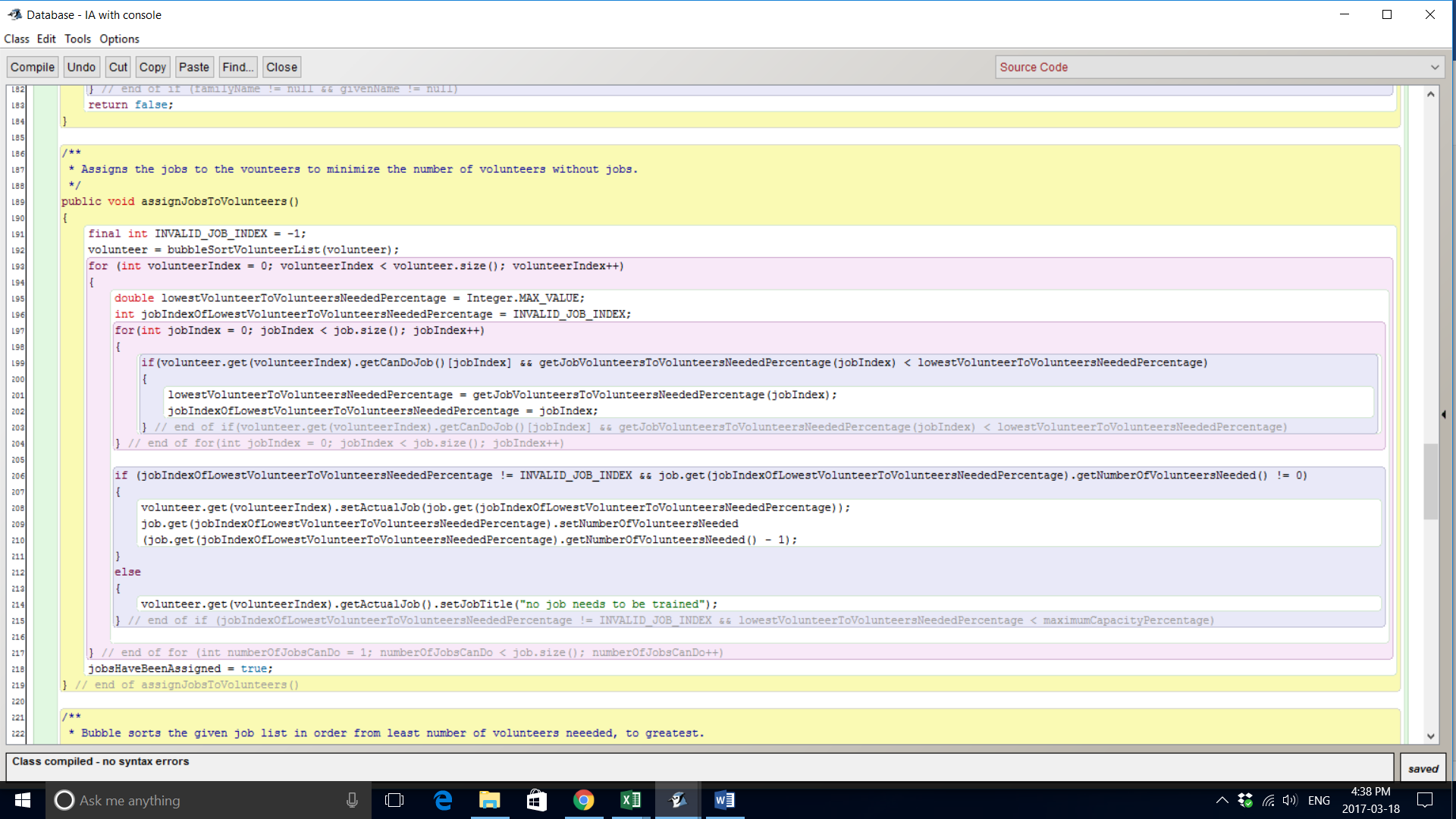




The bubbleSortJobList method sorts the job list in terms of number of volunteers needed. This allows easier traversal in other methods. For example, the displayExtraJobs method displays the jobs that still need volunteers in order from jobs that need the most volunteers, to jobs that need the least number of volunteers. Without a sorting algorithm, the linear search would have to traverse through the whole job list multiple times to iteratively print out the jobs and volunteers needed in terms of priority. With the bubble sort with complexity O(n2), the console simply has to traverse linearly through the job list once to print out the jobs and their volunteers needed in terms of priority.

## Algorithmic Thinking

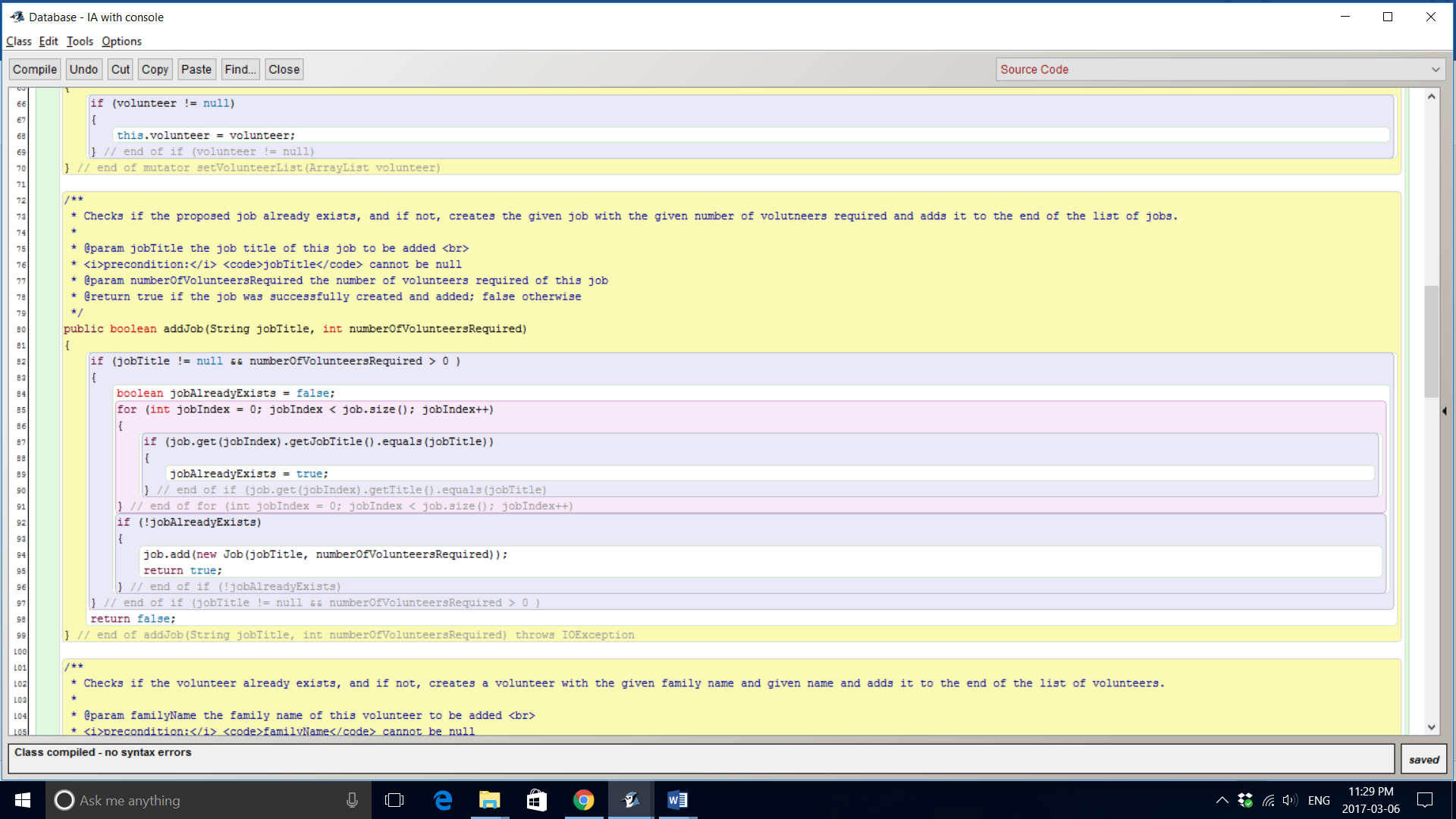
The most brilliant aspect of the solution is its algorithm for assigning jobs to volunteers to minimize the number of volunteers that don’t have jobs. The complete algorithm is shown below.



This algorithm uses the multiple nested for loops and conditional statements to best assign jobs to volunteers. The nesting of loops and conditional statements within each other iteratively check through each list to select the least flexible volunteers and jobs that are least popular so that those volunteers and jobs are assigned first. Jobs are iteratively assigned so that the database maximizes the number of volunteers with jobs. This algorithm uses a bubble sort to sort the volunteer list in terms of the job flexibility of volunteers, to once again improve the efficiency of the algorithm.

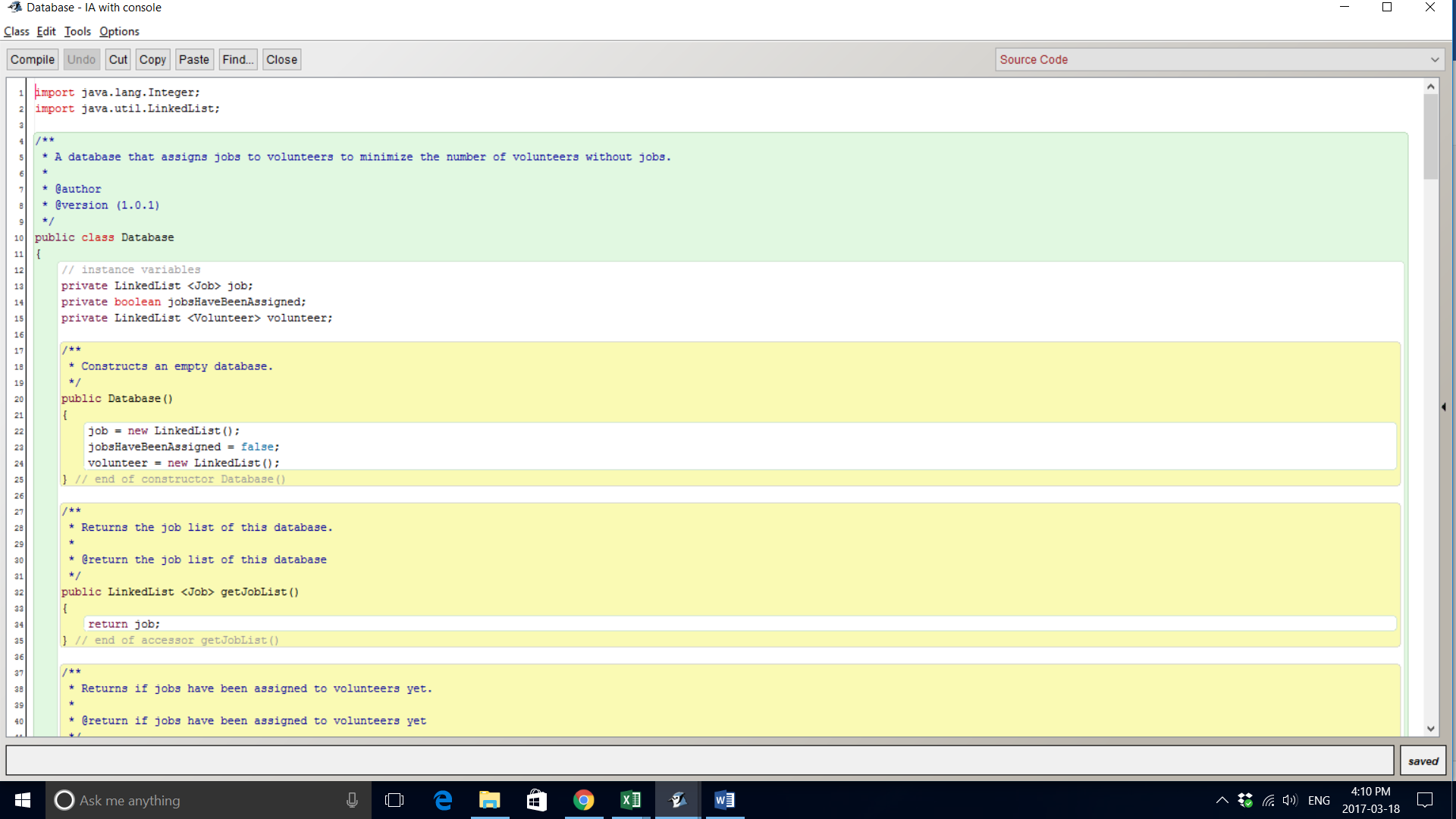
## Java Console Input/Output Skills

Databases also have to fend off the problem of input error. The database limits input error in several different ways. For example, if the user tries to input a job title that he already inputted, the database will disregard the input and print out an error message. This is done by iteratively checking through the job list and seeing if the job title string matches with the new job title.



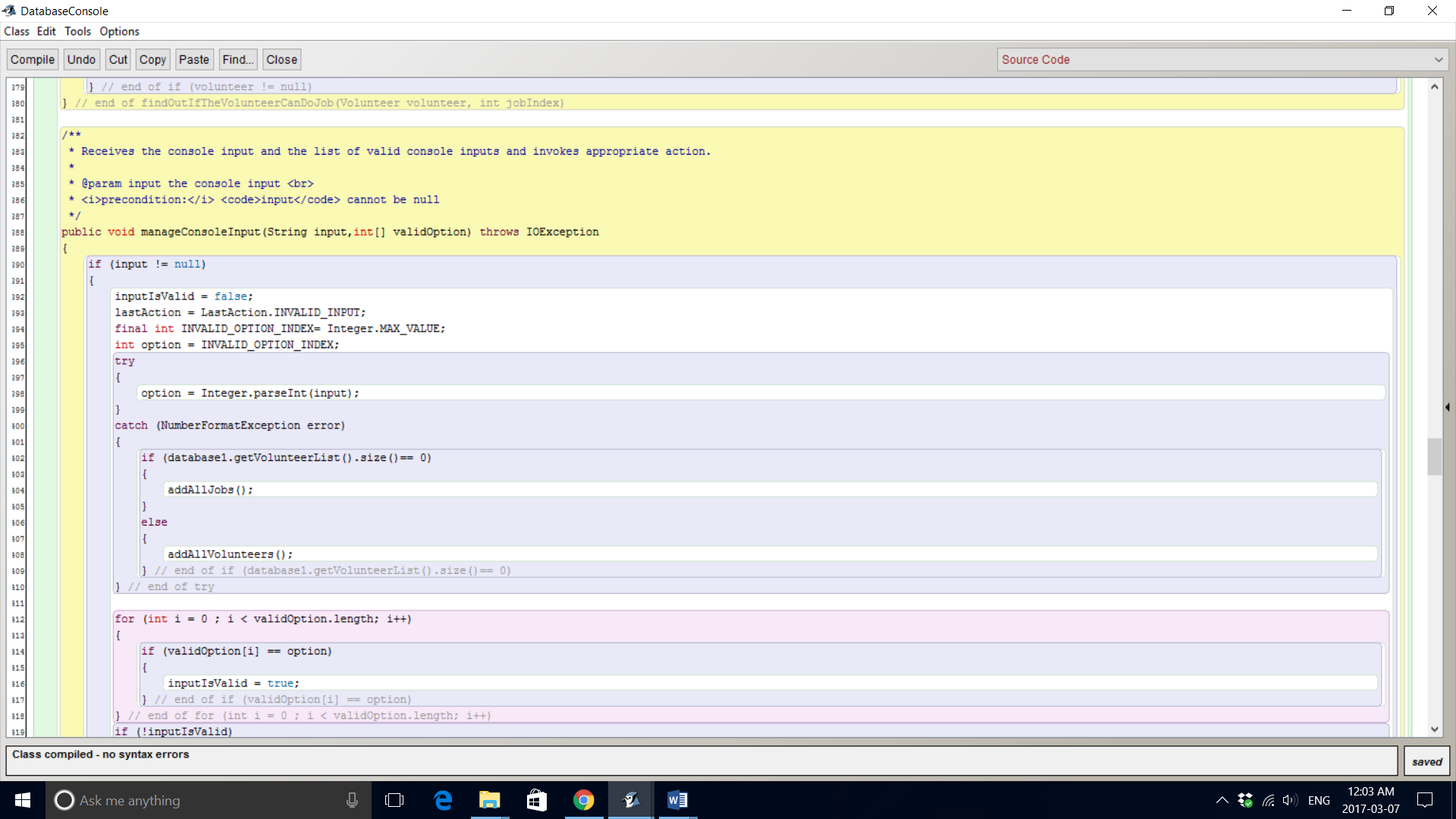
The database does the same thing with volunteers that already have their names’ in the database.

## Dynamic Lists



Like most databases, the solution uses dynamic lists. The database employs linked lists for both the volunteer and job list. Dynamic lists are necessary as their expandability allows for scaling number of jobs and volunteers. Arrays would be inefficient as their static sizes disallow changing number of volunteers and jobs which are against the use of databases. The solution uses linked lists over array lists as deletions and insertions are common actions done to the lists, and linked lists are very efficient at deleting and inserting.

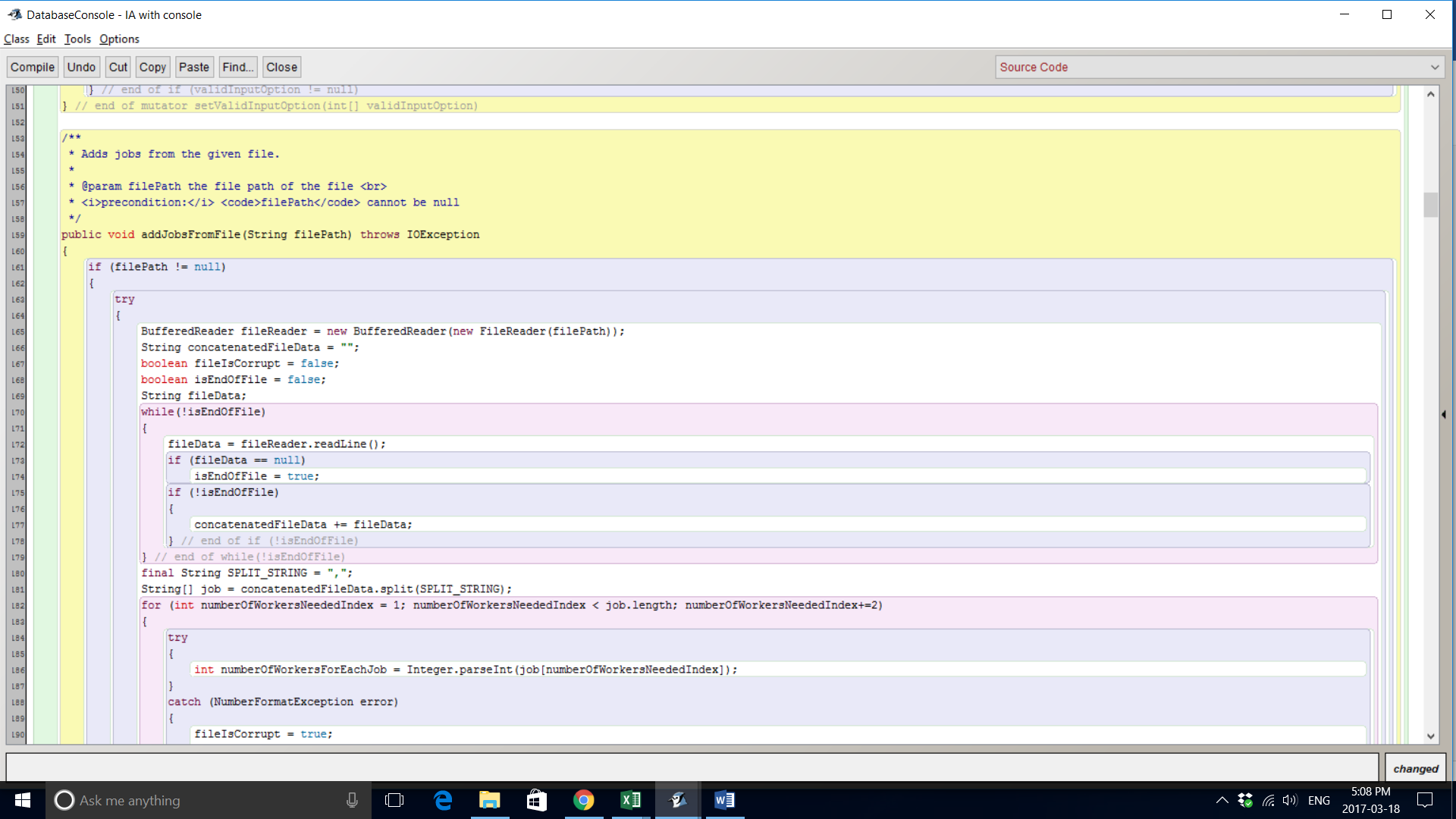
## . Error Control



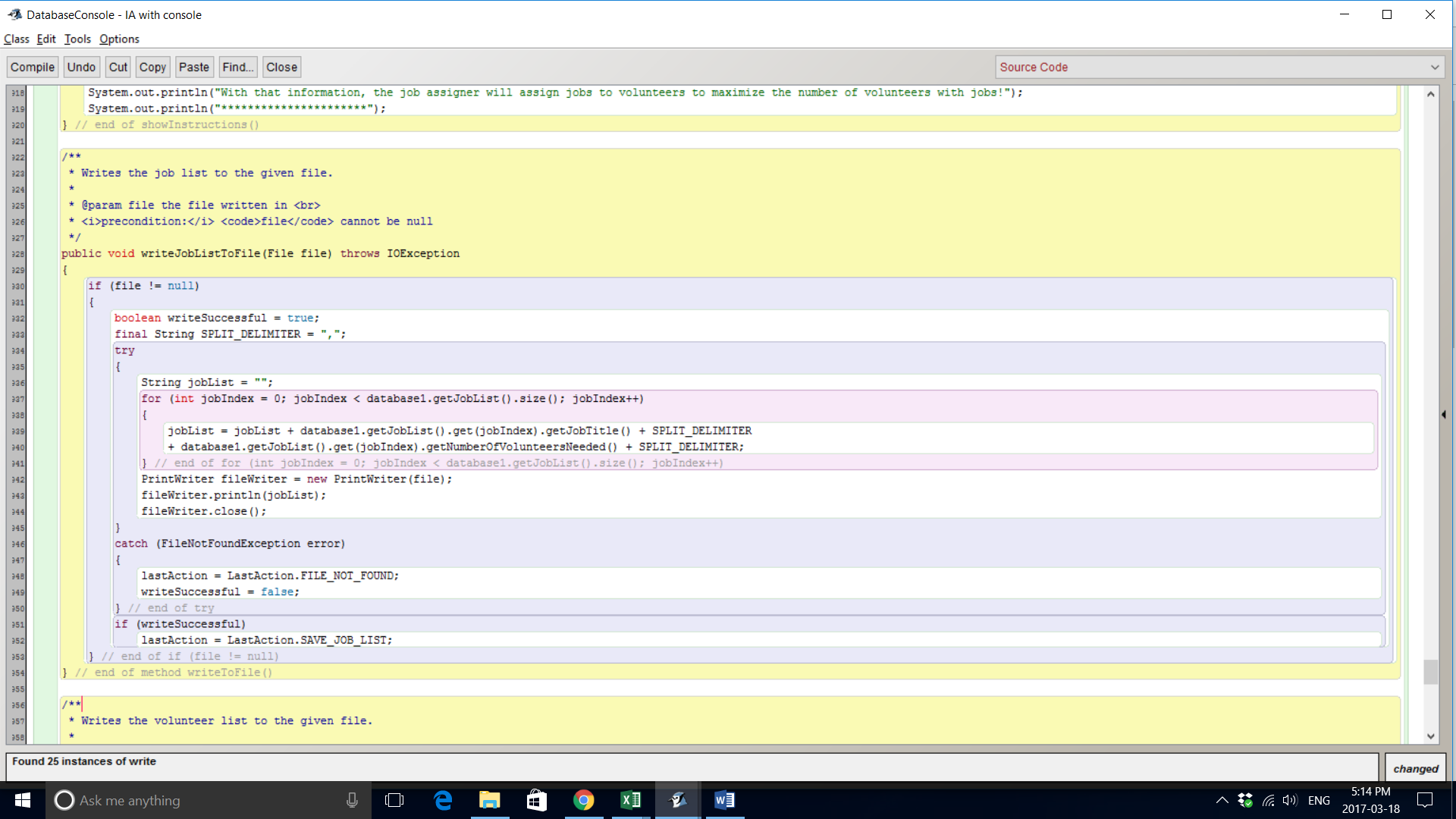
The selection of options on the menu relies on parsing the string taken into an integer. If the user does not type a string that can be parsed into an integer, a NumberFormatException occurs. Thus, a try-catch statement is necessary to appropriately react to the invalid input. By catching the error instead of throwing it, the console can inform the user of the issue so that it does not happen again.

## File I/O

The database can read from files. The database is able to read from comma-separated values files to import and export volunteers and jobs. Comma-separated values are elements that are separated by commas. The database reads the entirety of the file, and breaks down the string into substrings in an array. It then creates the volunteer/job based on the elements in the array. This is shown below.



The database can write to the volunteer and job lists to files. It iteratively takes each volunteer/job and writes their instance fields to files to create a CSV file. This file I/O allows Joel to save his work as the CSV files can be read again later for further edits.



Word Count: 712