# CSE 1322L - Assignment 6 (Spring 2026)

## Introduction

KSU students have access to a service called Steppingblocks, which can be accessed here. This service has many features, such as a job board, a tuition calculator, a personality quiz to help determine one's career and major, as well as aggregate data on KSU alumni: which industries they are currently working in, how much they make, their job titles, and more. In this assignment, you will write a program which reads some of that aggregate data, and then generates a report on it.

## Comma-Separated Files

In this assignment, you are provided with a csv file, which is a comma-separate-value file. This is nothing more than a text file where every line is an entry and each entry has several fields which are separated by commas. Below is an example of the contents of a csv file:

```
name,age,address
Alice,30,1st Street
Bob,40,2nd Street
Charlie,50,3rd Street
David,60,4th Street
```

The example above has 4 entries, and each entry has 3 fields: name, age, and address. Note that the fields are separated by commas and that the first line usually contains the field headers (i.e.: the field names).

The file you will be using contains 16 fields total. See below their descriptions:

**city**, and **state**: the city and the state where the alumnus currently works

**target_university**, **target_edu_level**, **target_edu_year**, **target_edu_major**, and **target_edu_college**: "target_university" always lists "kennesaw state university". The other fields list what education level the alumnus achieved (e.g.: bachelor's), when they achieved it, what was their major, and their college.

**highest_edu_university**, **highest_edu_level**, **highest_edu_year**, and **highest_edu_college**: highest level of education the alumnus achieved, which year they achieved it, and at which college and university. Note that this can be different to the "target" fields: a student who got their bachelor's at KSU but their master's at UGA will have "KSU" listed under "target_university" but "UGA" under "highest_edu_university".

**current_job_category**: current job category of the alumnus, such as "management", "logistics", "research", etc.

**current_employer**, **current_employer_start_year**, and **current_employer_industry**: name of the alumnus' employer, year when they started working there, and the employer's industry (e.g.: "telecom", "retail", "education")

**estimated_salary**: how much the alumnus makes per year

Note that not all fields will be populated for all entries: line 5 lists someone who works in Marietta and got a bachelor's in 2004, but it doesn't list their major. While your program must read all entries in the file, entries with missing information must be discarded. A salary of $0 will be considered missing information.

This assignment provides 3 files: "final-data.csv", which contains the data you will read, "report.txt", which contains the report your program must generate, and "errors.txt", which will contain all the entries with missing fields, the name of the missing field, and in which line said entry can be found in "final-data.csv". Be sure to check the first few lines of all files so you have an idea of what the data is and what the program must output.

## Requirements

Download the zip file that is near these instructions and place "final-data.csv" into your project's root folder.

Create a class called IncompleteEntryException with the following properties:

- Is a subclass of Exception
- Has an integer field called "lineNumber" and a String field called "info"
- **IncompleteEntryException(String, int, String)**: This constructor passes its first argument, the error message, to its superclass constructor. It then assigns the other two arguments to the object's fields
- Both fields have getters

Create a class called GraduateInfo, which will be used to hold entry data in "final-data.csv". This class has **16 fields**. The first line of "final-data.csv" contains the names of all the fields. All fields must be **public** and of type String except for target_edu_year, highest_edu_year, current_employer_start_year, and estimated_salary; these fields must be of type int.

Create a class called Queries. This class will contain only static methods. These methods will be fed an arraylist of GraduateInfo and return some data based on the information in the arraylist:

- **static int higherEdAtDifferentUni(ArrayList<GraduateInfo>)**: This method returns the number of graduates in the argument which got their first degree at KSU, but got their highest degree elsewhere.
- **static int[] highestEducationLevel (ArrayList<GraduateInfo>):** This method counts and returns the highest education level of all the graduates in the argument. The returned array must have a length of 5, where its indices must store:

    - **[0]:** number of graduates whose highest education level is "certificate"
    - **[1]:** same as above, but "associates"
    - **[2]:** same as above, but "bachelors"
    - **[3]:** same as above, but "masters"
    - **[4]:** same as above, but "doctorate"
- **static int[] lastYearAtKSU(ArrayList<GraduateInfo>):** This method counts and returns the number of KSU graduates per period. The returned array must have a length of 5, with each index storing the number of KSU alumni that graduated in that period. The periods are as follows:
    - **[0]:** Earlier than 2005
    - **[1]:** 2005-2009
    - **[2]:** 2010-2014
    - **[3]:** 2015-2029
    - **[4]:** 2020 or later

  **Note**: you must count only the latest graduation year at KSU. If an alumnus got their bachelor's at KSU in 2007 but their master's at UGA in 2017, they must only be counted under index 1. Otherwise, if they got both their bachelor's and their master's at KSU in the years above, they must only be counted under index 3.

- **static ArrayList<String> topFiveIndustries(ArrayList<GraduateInfo>):** This method returns an arraylist of strings with exactly 5 elements. The elements are in the following format:

  ```
  {industry}: {count}
  ```

  Where {industry} is a value seen under "current_employer_industry" and {count} is the number of alumni working in that industry. Note that this method must return the **top 5 industries** which employ the most alumni.

In your driver, do the following:

- Prompt the user for a file name
- If a file with that name does not exist, print an error message and terminate the program
    - Your program must do this by explicitly catching a FileNotFoundException
- Otherwise, create an ArrayList of GraduateInfo. Then, open the file and read it line by line
- Read every line as a string, split() said string using comma (",") as the delimiter, and then determine if the line is a valid entry or not.
- If an entry is invalid, do the following:
    - Throw an IncompleteEntryException with the following information:
        - The error message "Missing field '{field}'", where {field} is the name of the first field from the left that is empty
        - The line number where the entry was read from
        - The whole entry, verbatim

    - Catch the exception above **explicitly**.
    - Write to a file named "errors.txt" the contents of the exception in the following format:

      ```
      {error message}, line {line number}
      {whole entry}
      ----
      ```

    - see the provided "error.txt" for an example of what your file should look like
- If an entry is valid, create a GraduateInfo object using the line's information, then append the object to the arraylist.
- Once the file has been completely read, create a file called "report.txt". Call the Queries methods in the order below and write their outputs to "report.txt"
    - **higherEdAtDifferentUni()**
    - **highestEducationLevel()**
    - **lastYearAtKSU()**
    - **topFiveIndustries()**
    - see the provided "report.txt" as an example of what your file should look like

## Deliverables

- IncompleteEntryException.java
- GraduateInfo.java
- Queries.java
- Assignment6.java (driver)

## Considerations

- You will get partial credit for partial work, as long as the rubric permits it.
- You will have to make use of String methods to complete this assignment. You can find their documentation in the links below:
    - https://docs.oracle.com/en/java/javase/23/docs/api/java.base/java/lang/String.html
- You will also need to use Arraylist methods:
    - https://docs.oracle.com/en/java/javase/23/docs/api/java.base/java/util/ArrayList.html
- There is no need to submit "report.txt" or "errors.txt" as your program should generate those when run. There is also no need to submit "final-data.csv", as your grader already has this file.
- Most of this program's input comes from the provided csv file, and most of this program's output is in either "report.txt" or "errors.txt". Be sure to inspect all these files so you can have a good idea as to how the program is behaving.
- The methods in Queries have been described in ascending order of difficulty.
- Writing a constructor for GraduateInfo will be very tedious. Once you've declared the fields, you should use IntelliJ's "Generate" feature by right-clicking the class

body, selecting "Generate", then "Constructor", selecting all fields, and pressing "ok"

- Likewise, there is a better way to figure out which field is missing when printing to "error.txt" instead of just writing a big IF block.
- A good way to implement topFiveIndustries() is by using a HashMap to keep track of things. Unfortunately, we won't see those until a future module so, for now, you'll have to keep track of how many alumni each industry has in some other way.
- When getting data from an outside source for bulk processing, the data usually needs to be "cleaned".
    - irrelevant fields should be dropped, and values should be standardized (e.g.: "vp" and "Vice President" should both be replaced with "vice-president")
- Do you think you can write a Query method that determines the top 5 industries with the best salary on average? Feel free to create any helper methods or helper classes to aid you.

## Sample Output (user input in red)

```
[KSU Graduate Job Report]
Enter file name: data.csv
Could not find the specified file.
Program complete.
```

## Sample Output (user input in red)

```
[KSU Graduate Job Report]
Enter file name: final-data.csv
There was a total of 4278 valid entries and 5722 invalid entries.
Report written to 'report.txt'. Errors written to 'errors.txt'
Program complete.
```