This Java code implements a program that processes data from a CSV file containing app-related information. The code segments within the Main class preform various operations such as counting the number of apps per category, identifying top companies and developers, managing budgets and segregating downloads into free and paid categories.

In Main 4 HashMaps were made along with an ArrayList to help organize and manage the data needed.

The method NumOfAppsPerCategory counts the number of apps in each category and writes the result to a CSV file named AppsPerCategory.cvs. It utilizes a HashMap to store the count for each category.

Top100Companies indentifies the top 100 companies based on the number of apps they have released. The results are stored in a CSV file Top100Companies.csv. It utilizes stream processing and sorting to achieve this.

Top3Developers finds the top 3 developers based on the number of apps they have developed, excluding the apps associated with companies. The results are written to the Top3Developers.csv CSV file.

In the Budget method we can see the number of applications that can be bought with each of the budgets, and the results are written in the NumberOfAppsPerBudget.csv file.

FreeNPaidDownloads categorizes app downloads into free and paid, storing the results in the FreeNPaidDownloads.csv file. It uses a HashMap to track the number of downloads for each category.

The SplitLine function has a crucial role in parsing each line of the CSV file. Its purpose is to split a line into individual values, handling cases such as where values may be enclosed in double quotes and containing commas within those quotes. This part of the code was obtained from my colleague Tarik Bašić.

Through various methods, the code encapsulates critical sections within try – catch blocks to gracefully handle exceptions and mantain program stability.

Scanners are used to parse through the contents of the CSV file, using methods like hasNextLine(), nextLine() to navigate and extract data line by line.

The FileWriter class is used to create and write data into output files. FileWriter manages the writing process, allowing the code to organize and store data in the specified output files.

The use of iterators is seen during the output generation. They provide a way to loop through the key – value pairs stored in the HashMaps.

Though it is important to note that the FreeNPaidDownloads and Budget functions have potential issues that affects the final output of the files they are intended for. The FreeNPaidDownloads function encounters problems in categorizing downloads into free and paid apps, while the Budget function I believe is innacurate with its final output on how many apps can be bought with the two budgets provided.

- What was most challenging?
  The most challenging part was figuring out the proper regex for spliting the lines properly, even turning to ChatGPT for help, but with the help of my colleague it got finished in the end.

- What you expected to be very hard, but it turned to be easy work for you? (if anything)
  I expected the task to categorize apps per company to be harder than it actually was. It still was not simple but simpler than I had imagened.

- What you expected to be simple, but turned out to be very complex? (if anything)
  I expected the tasks for free and paid downloads to be simple but in the end, not being able to get the correct output at all, proved me wrong.

- If you are starting assignment again, what would you do differently
  Now that I look back at my code, I would have liked if I created sepperate functions for finding and writing out data and even reusable functions for certain data to give the code better readability.