Initial input:

In this section, the program would take in command-line arguments for the events and statistics files to be read, as well as the number of days to generate activity logs for. These information are stored in local string variables eventsFile, statsFile and local integer variable days respectively. The events file is then being read first, and due to the format of the files a split() function is executed to separate and store each data item in a string array. Each array would represent an event, and in the case where there is no specified maximum limit, a placeholder large integer 99999 is inserted to facilitate random number generation in the activity engine. Each string array would then be added to a global ArrayList<String> named eventsArr. The same steps are conducted when reading the statistics file, and the extracted data is added into global ArrayList<String> statsArr.

One potential inconsistency in data is when the listed numbers of events and statistics in each file do not match. When that occurs, the program would throw an exception specifying that the data array sizes do not match. Another inconsistency is when the list of event names in the events and statistic file do not match, and similarly this would throw an exception stating the mismatch in labels. Other noted possible inconsistencies which the program does not handle include a possibility where the generated number of logins is 0, however other events are generating values more than 0, and that the number of events read in do not match the number stated in the first line of the file.

Activity Engine and Logs

To achieve random generation of numbers close to a given mean and standard deviation, the Math function nextGaussian() is implemented. nextGaussian by itself generates a double between 0 and 1, and using the formula (nextGaussian x standard deviation + mean), the resulting number is now a random number that follows a specific mean and standard deviation. This value also forms the z-score. The saved value could be an integer if the event is discrete, or a double rounded off to 2 decimal places if the event is continuous.

The data for the log file will be written into “Activity.txt” which is a simple human-readable text document. Internally in the program, the contents would be saved in a nested array list (ArrayList<ArrayList<String>>) named log for future usage in subsequent engines, and also saves the need for re-reading the data before further processing.

Analysis Engine

The analysed statistics will be written into “Analysis.txt”, same as the file format used for Activity.txt. The system internally stores all computed mean and standard deviation values in ArrayList<Double> named meanArr and stdevArr respectively. These arrays reset itself with each time the analysis engine is run to accept new sets of mean and standard deviation values.

Alert Engine

In this section, the alert engine is run under a while loop where the program would always prompt for input until the user enters ‘q’ to quit. Upon entering the new statistics file name and number of days, the new statistic file will be read, then the activity and analysis engine would be called to process the new data. The new activity and analysis logs will be saved as “Activity2.txt” and “Analysis2.txt” respectively, and another file “Anomaly.txt” is saved with logs on each generated day’s anomaly count. In the terminal, the program would also print a line for each day to indicate whether that day has exceeded the calculated threshold.