CSCI262 Assignment 3

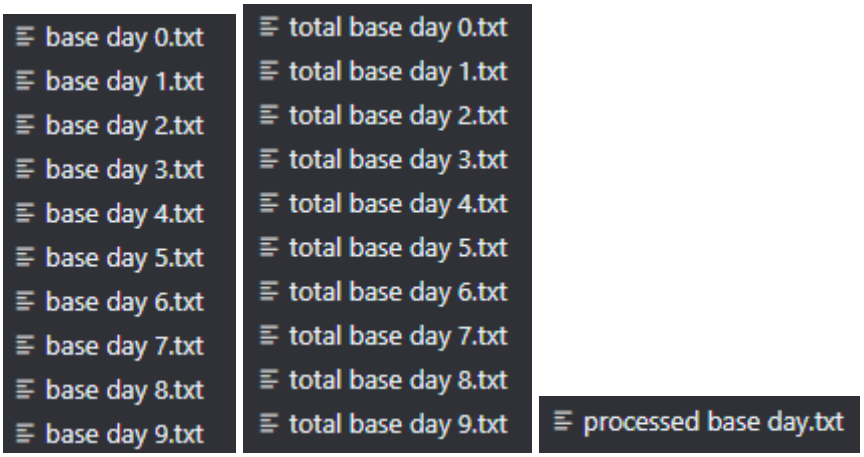
Full Name	UOW ID
Chng Yia Qing Whitney	6956865
Ching Jun Hao	6958709
Junior Tantono	6957201

Initial Input

Upon starting the program, the user has to specify the number of days of logs along with the main java program and the text files that are going to be processed.

```
PS C:\Users\PC\Documents\SIM\CSCI262\Assignment 3\Test> java IDS2 Events.txt Stats.txt 10
Generating Events
Done
Analyzing Events
Done
Enter 'q' to quit or Enter TestStats Days:
```

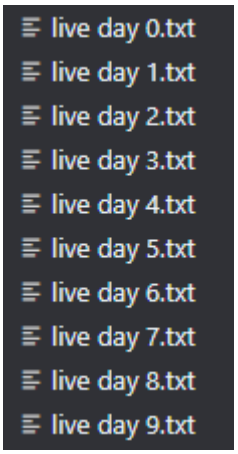
The program will then process both .txt files which will output a series of .txt files containing information about the logs. In this case, we specified 10 days which will produce these files:



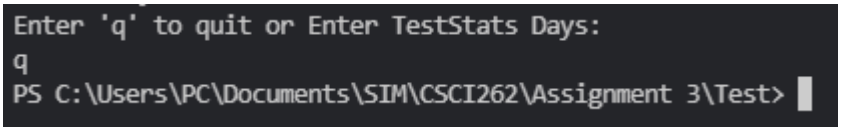

```
Enter 'q' to quit or Enter TestStats Days:
TestStats.txt 10
Generating Events ...
Done
Starting alert analysis, anomaly treshold 2880.0
No Anomaly detected
Enter 'q' to quit or Enter TestStats Days:
```

Subsequently, the user can either enter “q” to quit the program or provide another .txt file for the program to compare the TestStats with the base logs.

This will then produce another series of files:

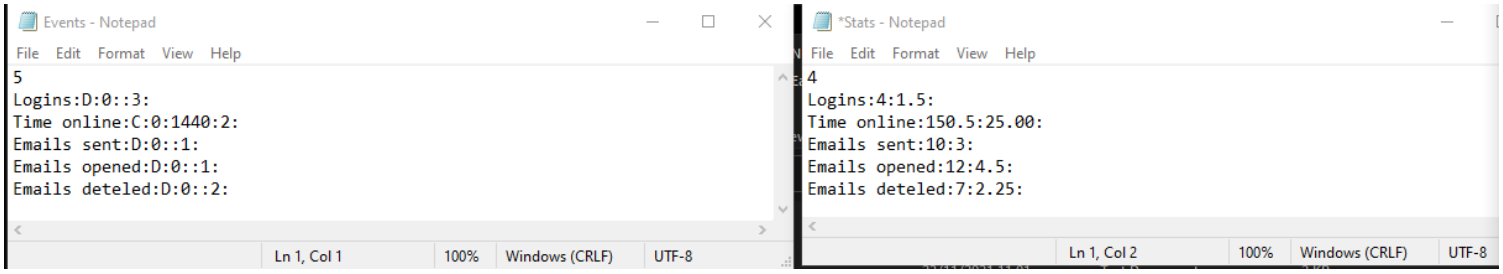


If the user wishes to exit the program, the input required will be “q”.

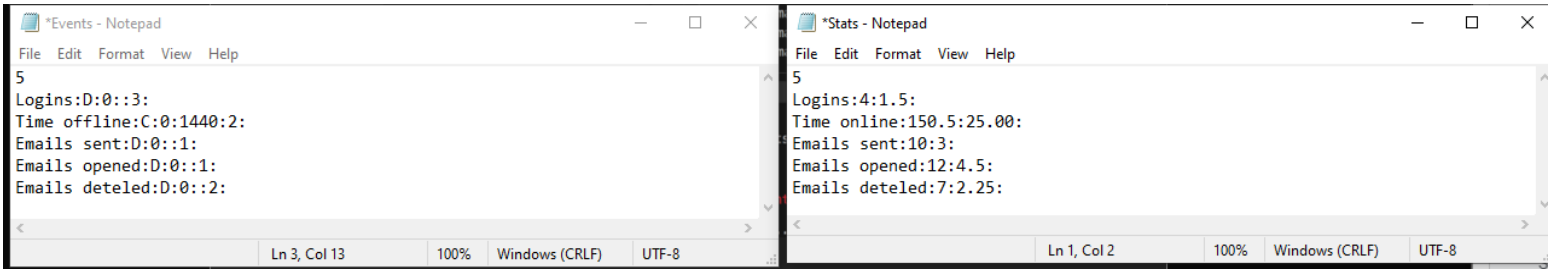


Potential errors that can be found while validating the consistencies between Events.txt and Stats.txt:

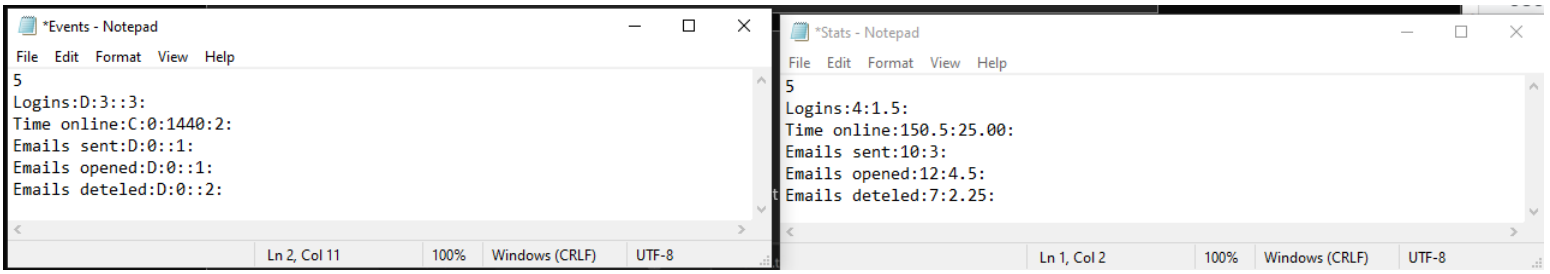
- Different number of monitored events



- Different event name or sequence (e.g Time Offline instead of Time Online)



- Minimum out of range (e.g Minimum value in Events.txt lesser than Stats.txt)



Activity Engine

When the program takes in the two base files provided by the user, the activity engine will then:

- generate timestamps for the logs
- process and output logs

The logs are generated by taking in the number of days specified by the user which will be put through a loop which will subsequently create new .txt files and print the necessary information inside each file.

There are two cases that differ between continuous and discrete events in this loop:

1) If the method is trying to generate information for Time online (Continuous)
    - a randomly generated double value for time online
    - The output of this event will be:
        time:eventName:doubleValue

```
if(eventStats[j].getType() == 'C')
{

    double value = eventStats[j].getRandValue();
    eventLog.add(generateTimestamp() + ":"
    + eventStats[j].getName() + ":"
    + String.format("%.2f", value));
```

2) If the method is trying to generate information for Logins, Emails Sent/Opened/Deleted (Discrete)
    - a randomly generated int value which decides how many times an event will happen
    - The output of this string of information will be:
        time:eventName:value

```
else if(eventStats[j].getType() == 'D')
{
    int events = (int)eventStats[j].getRandValue();

    for(int k = 0; k < events; k++)
        eventLog.add(generateTimestamp() + ":" + eventStats[j].getName() + ":" + 1);
}
```

Each variable is separated by ":".

base day 0.txt

```
04-09-28:Logins:1
09-24-39:Logins:1
19-26-10:Logins:1
13-28-25:Time online:105.49
19-10-07:Emails sent:1
05-40-35:Emails sent:1
04-39-37:Emails sent:1
09-09-43:Emails sent:1
23-34-40:Emails sent:1
06-08-20:Emails sent:1
04-14-21:Emails sent:1
08-04-04:Emails sent:1
07-16-27:Emails sent:1
19-16-41:Emails sent:1
12-00-54:Emails sent:1
05-14-16:Emails sent:1
23-11-23:Emails opened:1
20-33-43:Emails opened:1
07-54-40:Emails opened:1
04-32-33:Emails opened:1
13-20-49:Emails opened:1
10-36-48:Emails opened:1
18-15-03:Emails opened:1
21-15-42:Emails opened:1
11-09-13:Emails opened:1
13-26-11:Emails opened:1
03-44-17:Emails opened:1
13-31-26:Emails opened:1
17-36-43:Emails deteled:1
03-33-22:Emails deteled:1
23-39-24:Emails deteled:1
```

The naming convention of these base files are:

```
//print each line to each file
PrintWriter output = new PrintWriter(new File(prefix + " " + i + ".txt"));
for(String line  : eventLog)
    output.println(line);
output.close();
```

base day 0.txt
base day 1.txt
base day 2.txt
base day 3.txt
base day 4.txt
base day 5.txt
base day 6.txt
base day 7.txt
base day 8.txt
base day 9.txt

Analysis Engine

When the activity engine has completed generating the logs, this engine will process these logs. The totals for each event is acquired by:

- Splitting each part using ":"
- Storing each part to its variable, string Event or double Value.
- Going through a for loop to count the number of re-occurrences for each event and returning the counter value.

Subsequently, the information acquired above will be printed out into another .txt file called total base day that represents the total for each day.

This is done by looping through the days specified at the start, which will generate a .txt file for each day that contains the data.

- For each file, the information will be printed as such:
  eventName:total

```
//print out even total for each day
PrintWriter output = new PrintWriter (new File ("total " + prefix + " " + i + ".txt"));
    for(int j = 0 ; j < eventStats.length; j++)
        output.printf ("%s:%.2f%n", eventStats[j].getName(), eventCounter[j]);
output.close();
```

total base day 0.txt

```
1    Logins:3.00
2    Time online:105.49
3    Emails sent:12.00
4    Emails opened:12.00
5    Emails deteled:3.00
```

After the totals for each day has been generated, the program will then take in all the data from the files and calculate the mean and standard deviation.

- Loop through and take in the total values from each day
- Calculate the mean and standard deviation and storing them into double mean and double std
- Each event will be printed as such:

  eventName:mean:std

```
PrintWriter totalOutput = new PrintWriter(new File("processed " + prefix + ".txt"));
for(int j = 0; j < eventStats.length; j++)
{
    //calculating mean & storing it
    double total = 0;
    for(int i = 0; i < days; i++)
        total = total + totalEventCounts.get(i)[j];

    double mean = total/days;
    processedEvents[j].setMean (mean);

    //calculate Standard Deviation & storing it
    double sumDiff = 0;
    for(int i=0; i < days; i++)
        sumDiff += (totalEventCounts.get(i)[j] - mean) * (totalEventCounts.get(i)[j] - mean);

    double std = Math.sqrt (sumDiff / days);
    processedEvents[j].setStd (std);
    // write the output in the file
    totalOutput.printf("%s:%.2f:%.2f%n",eventStats[j].getName(), mean ,std);
}
totalOutput.close();
```

processed base day.txt

```
1    Logins:3.90:1.04
2    Time online:143.29:27.52
3    Emails sent:9.40:2.84
4    Emails opened:13.60:4.63
5    Emails deteled:5.80:1.78
```

Alert Engine

User will be prompted to enter a new Stats file that contains new statistics and number of days.

The following are possible errors that may occur while validating user inputs:

Inputting the wrong file

- The text file entered is not found in the directory

```
Enter 'q' to quit or Enter TestStats Days:
Invalid.txt
Please enter the correct format: StatsFile.txt days
Enter 'q' to quit or Enter TestStats Days:
```

Invalid file extension

- File extension wrong, only .txt files allowed

```
Enter 'q' to quit or Enter TestStats Days:
Test.tx
Please enter the correct format: StatsFile.txt days
Enter 'q' to quit or Enter TestStats Days:
```

Storing Data

Once there are no errors in the input, the program will proceed by storing new statistics by creating an instance of the class EventType for each event and will be stored in an array. The ActivityEngine will then run to produce data for the number of days entered and will then be stored in the live day.txt

This is an example if the user entered “TestStats.txt 10” without any errors.

```
Enter 'q' to quit or Enter TestStats Days:
TestStats.txt 10
Generating Events ...
Done
Starting alert analysis, anomaly treshold 2880.0
No Anomaly detected
Enter 'q' to quit or Enter TestStats Days:
```

A Threshold is calculated using the formula 2*(Sum of Weights), where weights are from the Events.txt.
For each day, an anomaly counter of each event is computed by:

[abs(data-mean) x weight] / stdev]

After computing the anomaly counter, the program will then start to flag out anomalies. If the sum of the anomaly counter is greater or equal to the threshold, it will then be flagged as "Anomaly detected on Day X, anomaly weights = XXXX"

```
Starting alert analysis, anomaly treshold 2880.0
Anomaly detected on day 0, anomaly weights = 5809.85
Anomaly detected on day 1, anomaly weights = 6965.55
```

If no anomaly is found, it will just display:

```
Enter 'q' to quit or Enter TestStats Days:
TestStats.txt 10
Generating Events ...
Done
Starting alert analysis, anomaly treshold 2880.0
No Anomaly detected
Enter 'q' to quit or Enter TestStats Days:
```

The program will then ask for a new set of stat file to input or the user can enter q (not case sensitive) if they wish to quit the program.

The following is an example of how the program will look like if the user decide to quit

```
Enter 'q' to quit or Enter TestStats Days:
q

C:\Users\User\Desktop\Test>
```