**Project 3 – CocoPanther**

**Activity 1:**

**Logic Flow:**

1. The program takes a file name as input, checks if the file is present or not.
2. If the file is present, it calls a parseFile() function.
3. The parseFile() function reads the file and parses based on some keywords found in the file.
4. Then, it prints details such as number of files in the log, files with most users committing to it, earliest commit to a file, author with most commits in the log.
5. Also displays the individual file contents- that is, the filename, earliest commit, the name of author who has made the maximum commit in that file and the number of commits made by him.

**Cases handled:**

1. Checking for invalid or non existent file.
2. If a directory name is given as file name, that is also flagged as error.

**Activity 2:**

**Logic Flow:**

1. In activity 2, the steps 1 to 3 in activity 1 are performed.
2. Then, details such as filename, total number of revisions, earliest commit date and the last commit date are written to a .csv file.
3. The writing is done using OpenCSV.

**Cases handled:**

1. Checking for invalid or non existent file.
2. If a directory name is given as file name, that is also flagged as error.

**Activity 3:**

1. The activity 3 performs the same steps 1 to 3 as activity 1.
2. Then, it shows a menu to the user to select the time period – annual, half yearly or quarterly.
3. Based on the user selection, the program prompts the user to either enter a time period as a year or allow the system to generate the top 20% commits for ALL the years specified in the log file.
4. For quarterly time chunks, the year is divided into 4 periods and for half yearly time chunks, the year is divided into 2 periods.

**Scenarios where program may fail:**

1. In the program start, the user is prompted for an input for annual, half yearly, quarterly or exit the program. If any other option is entered, a default message is printed and the program exits.

**Other edge cases handled:**

1. If the user enters a bad year, it is flagged and the user is asked to re enter the year again.
2. Years lesser than 2001 and greater than 2015 are not allowed since the log file has been generated for the duration of 2001-2015.

**Applications of this project:**

The program is useful to parse any type of log file generated in organizations. Log files are large in size and volume and should be continually maintained in organizations for a form of traceability to code and other artifact changes. However, it is necessary that these log files be using the same format as used in the emacs.log file if this code is to be used. If any other format is used, then the parseFile() function should be changed for the program to execute correctly. The fileData class structure is also modelled to closely resemble the format of this emacs log file.

Usage of bufferedreader for reading the log file also ensures that the latency to process files does not increase much, and is a better alternative than storing the whole file in memory before processing.

Open CSV can be used to analyse any type of log file to identify trends in the log committing of individuals.