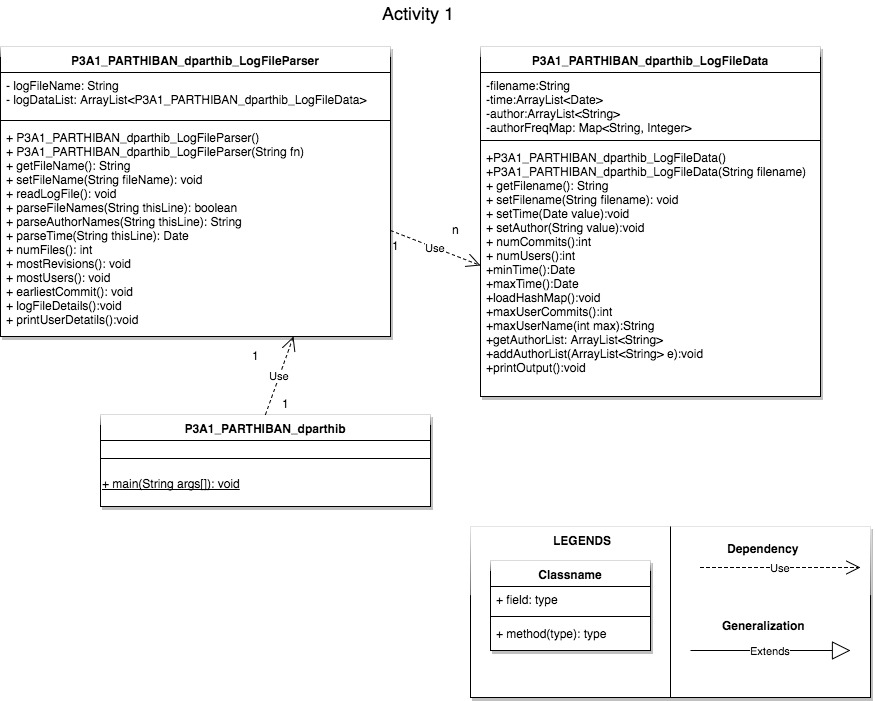
**[Project #3: CocoPanther!](https://blackboard.andrew.cmu.edu/webapps/assignment/uploadAssignment?content_id=_1279496_1&course_id=_8912816_1&assign_group_id=&mode=view)**

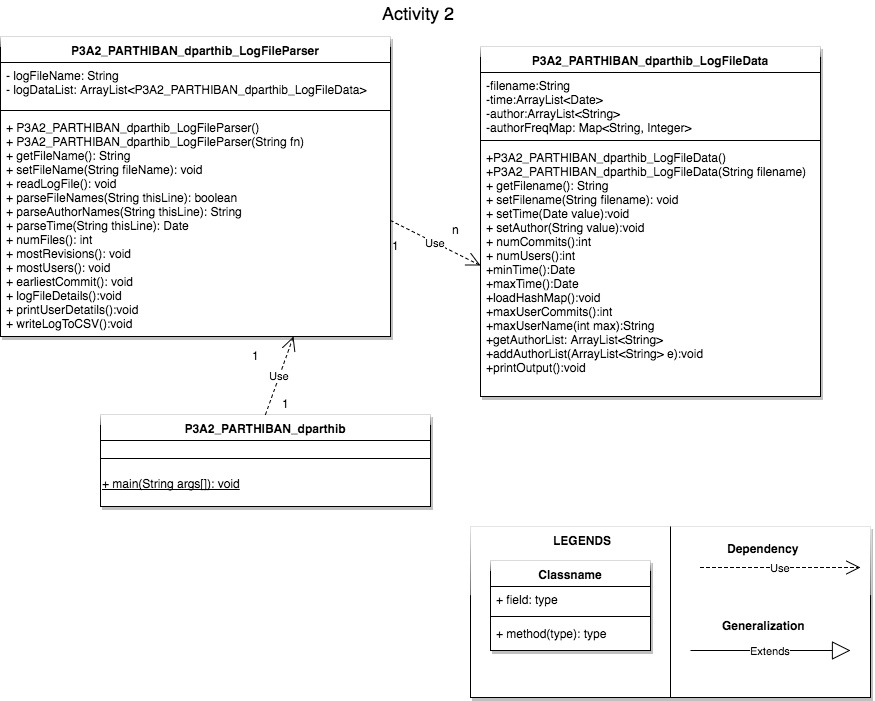
UML Diagram



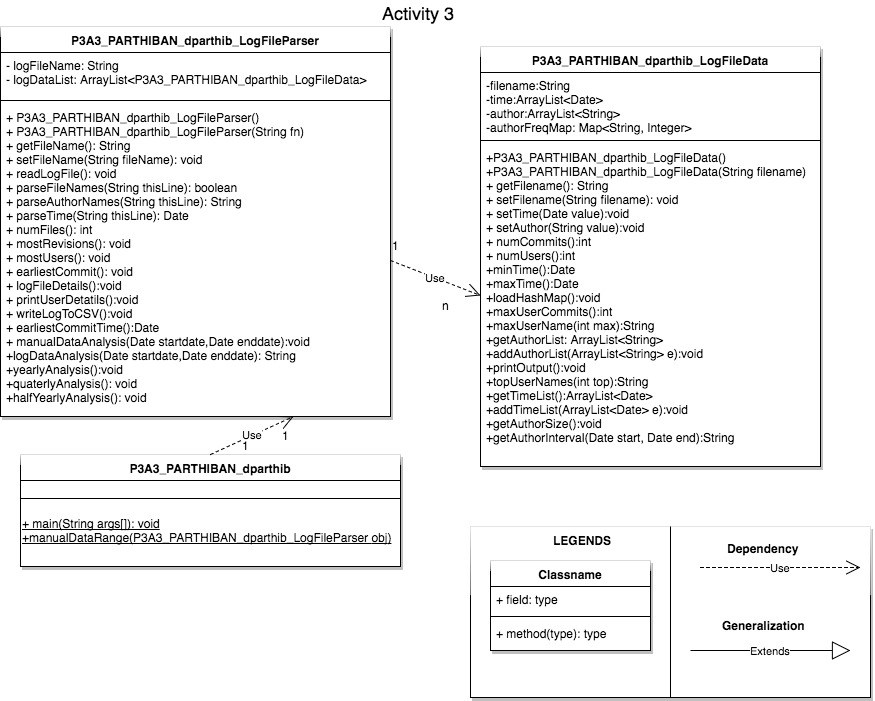
How the program works

The design choices for what goes into these three classes as listed below,

* P3A1\_PARTHIBAN\_dparthib – This is the class with the main method and only the logic to get the user input is kept here.
* P3A1\_PARTHIBAN\_dparthib\_LogFileParser – This is the class where it contains the logic for the total log file object and how to parse them. We create an object of the P3A1\_PARTHIBAN\_dparthib\_LogFileData for each revision file in the log file.
* P3A1\_PARTHIBAN\_dparthib\_LogFileData – This is the class where the individual files and their logic is stored.



The above rationale was again used to for Activity 2 where the modifications are at the main file and P3A1\_PARTHIBAN\_dparthib\_LogFileParser – additional log to work with CSV files (writeLogToCSV). Nothing else changed from Activity 1.



For activity 3 the modifications were at the below level,

* P3A3\_PARTHIBAN\_dparthib – This had new logic added to get input from the user for data analysis.
* P3A3\_PARTHIBAN\_dparthib\_LogFileParser – This had new methods like quaterlyAnalysis, halfYearlyAnalysis, yearlyAnalysis, manualAnalysis to generate the corresponding reports to CSV.
* P3A3\_PARTHIBAN\_dparthib\_LogFileData – new methods were added to generate the top 20% of the users.

Where can we Extend this Program?

Since we are actually parsing the log file of a version control system the classes LogFileParser & LogFileData can be extended to parse the data of similar systems like Git/Stack. Apart from these all software engineering applications produce a log file. Hence this program can be used to analyze them and generate reports.