1- if the issues tracker system is bugzilla then download the bugs of the project as csv file , the filter the issues via filter\_issues\_bugzilla.py , if the system is jira then download issues and rename the features to be the same as bugzilla via the script rename\_features , if the the issues are hosted in github issues then run the script fetch\_version\_github.py . those scripts will give the issues of the project filtered

2- clone the project and go to it then run the script tags.py which will create tags\_commits.csv

3- run the script szz\_vfinal with the specified input mentioned in the script (eg : python3 szz\_vfinal.py tags\_commits.csv "#\d+" https://github.com/eclipse/paho.mqtt.java.git fetch\_jira\_bugs/github/issues/paho.mqtt.java.json projects/paho.mqtt.java , where argument1 is the list of the tags and it’s correspond commit , argument 2 is the pattern of the issue used in the project , argument 3 is url git repo of the project , argument 4 is the list of issues and finally is the path of the project locally ) you have to create the jar file mentioned here <https://github.com/wogscpar/SZZUnleashed> also (this file and git\_log\_toarray should be in the same directory where you run the script)

the idea in this script is is in each release we search for the issues fixed and create issue\_list.json then execute the second phase of the szz algo (important note here that in the second phase we search for the commit that introduced the bug in all the project not just the release in which we found the issues fixes because an issue may had been introduced not int the same release as the fix)

bug\_cleaning\_data\_vfinal.py is the script that count the number of bug in a class per release .

(using cat command we merge all szz output of the project (we got multiple files because we run each release separately the file generated in each release is “fix\_and\_introducers\_pairs.json”))

we load all fix commit and commit introducing bug and keep them in order .

Then remove redundancy some fix commits associated with bug commit repeat so we remove redundancy .

Then for each commit introducing bug we search for it’s release and we check the files that had been modified in both fix commit and commit intro bug so we create a dict containing list of this classes found per release .

Finally we count the occurrence of each class (which will be the number of bug in that class) in release and create file of classes with number bugs for each release .