METHOD TRACE ANALYZER

TEAM NAME

Akatsuki

Jatin Jha

Sakar Jain

ROLE OF EACH TEAM MEMBER

Jatin Jha :-

*) Developed java code for generating log files for both passing case and failing case
*)Worked on comparing both files using generated log files to find out the anomaly
*)Used Xtrace commands to generate trace files and using trace format to convert those unreadable files into readable log files
*)Developed a java code to read log files line by line and compare them to find out that whether the exception exist or not
*) Developed GUI using Window Builder where user can input failing case and passing case file names
*) Compare methods of both classes to find out which method is taking extra time and represent it using bar graph with the help of jfree chart a java library
*) represent number of times each method called in tabular form

ROLE OF EACH TEAM MEMBER

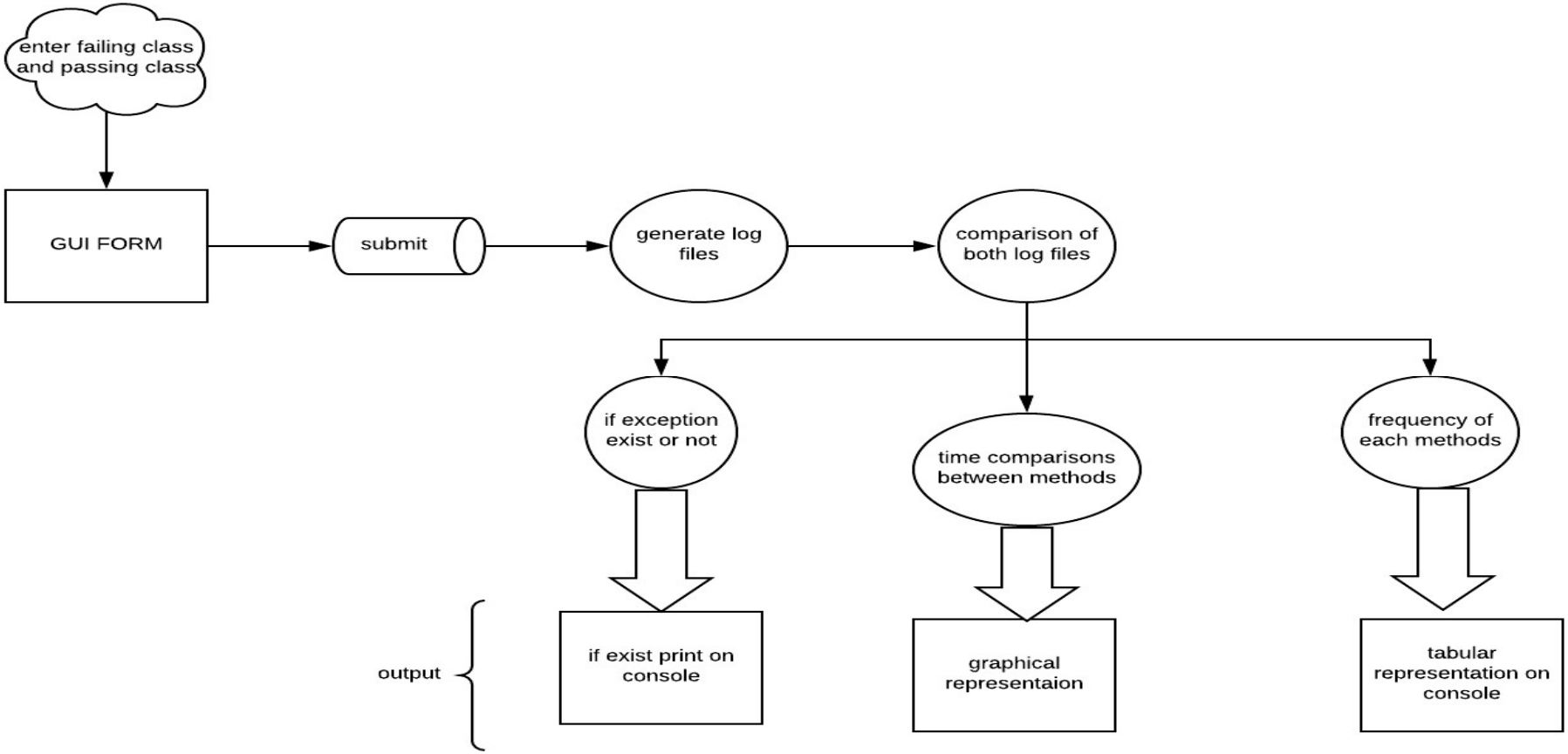
Sakar Jain :-

*)Integration of all modules into a single code
*)Suggested ideas and helped in implementing the module to detect exception /anomalies in the generated log files

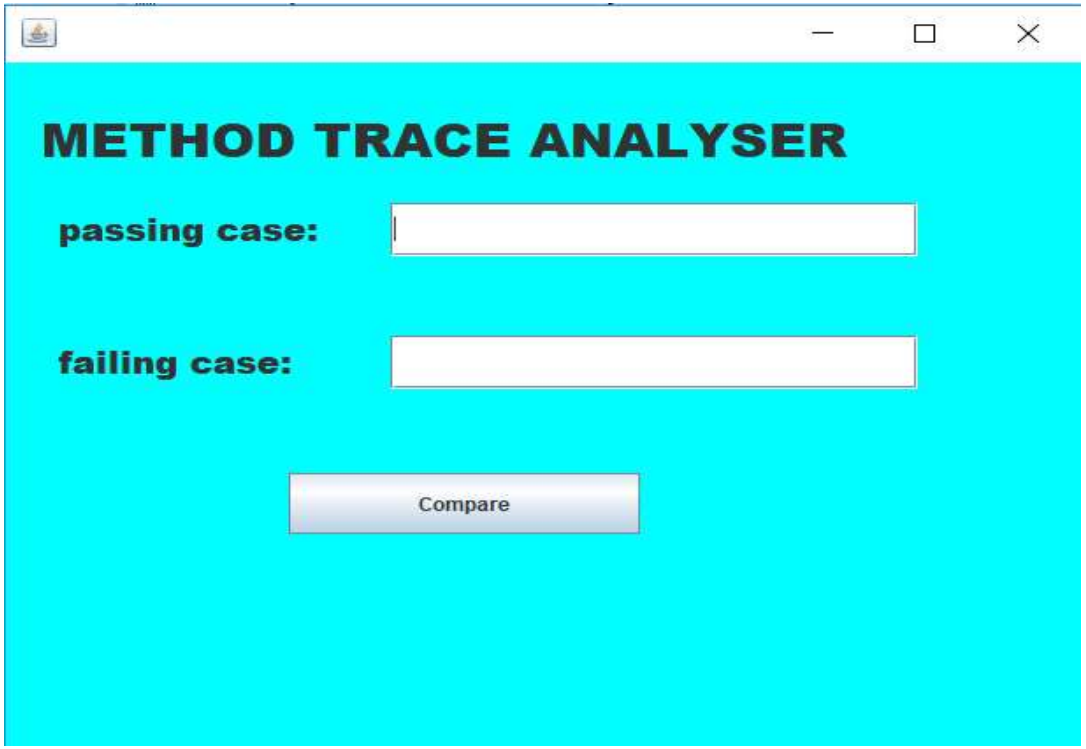
TECHNOLOGY STACK:

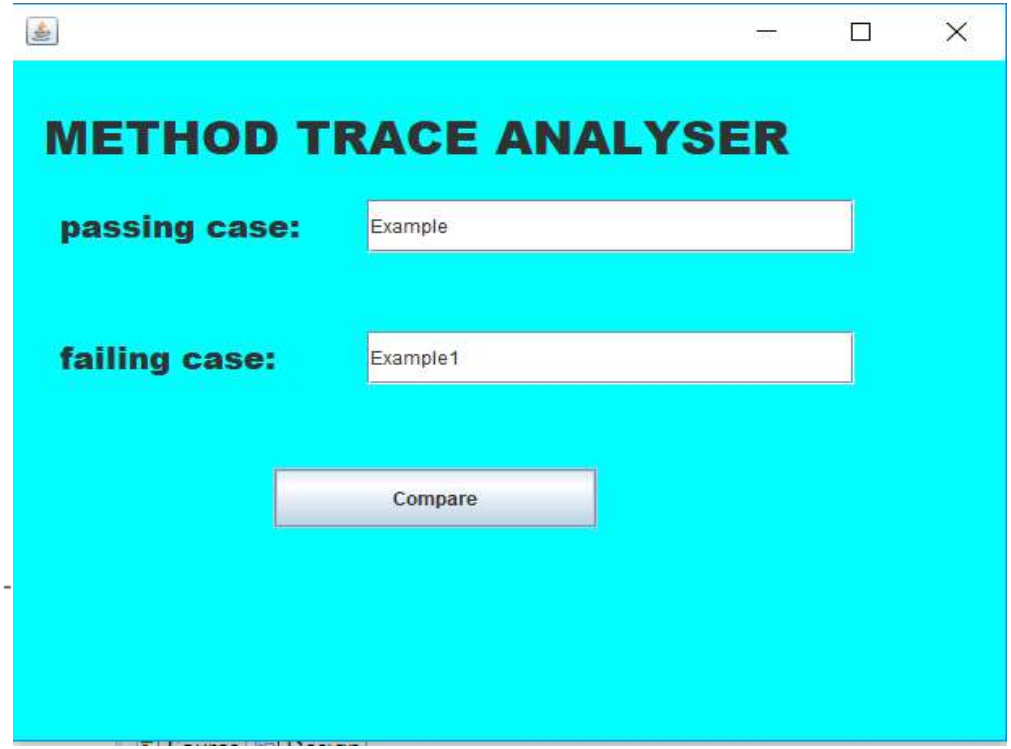
- Java
  - IBM java method trace
  - Eclipse
  - WindowBuilder
  - OpenJdk with Openj9
  - JfreeChart
- Github

FLOW CHART OF PROCESS



SCREENSHOT OF GUI





OUPUT ON CONSOLE

Tasks | Console | Servers | Error Log

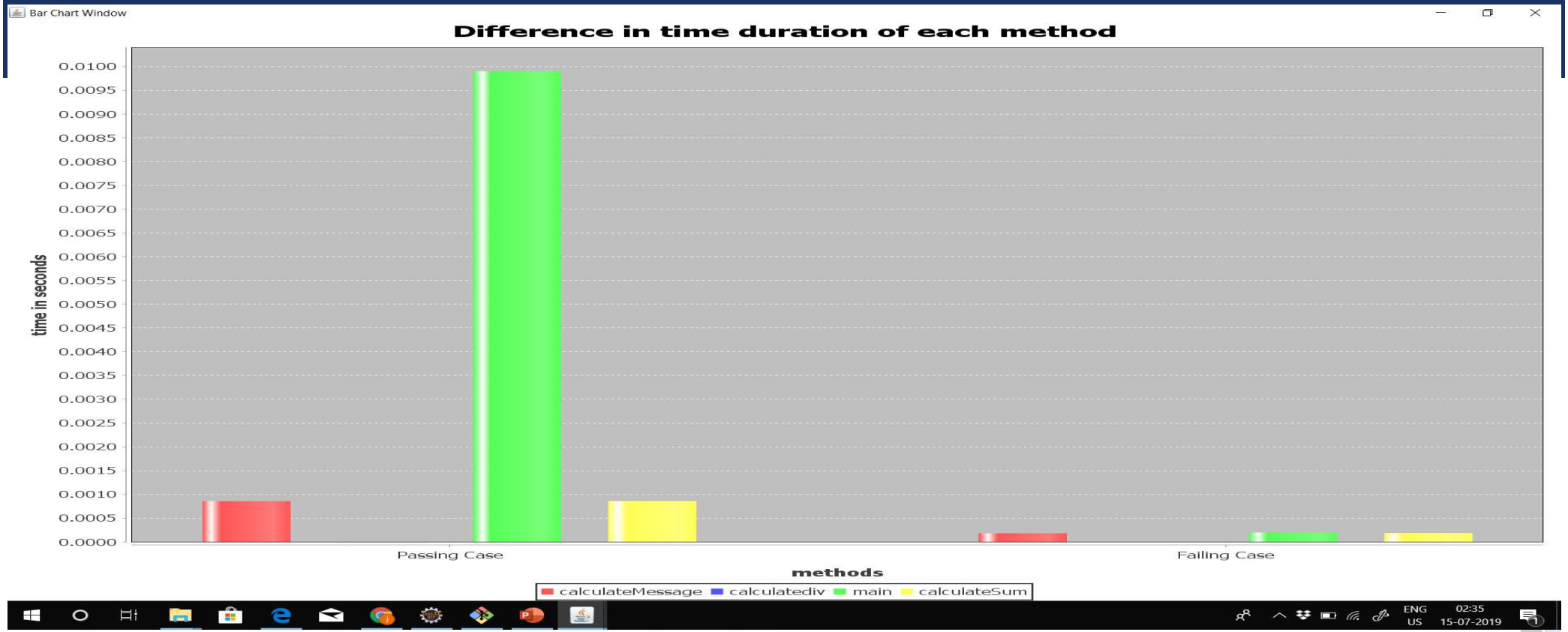
MethodTraceGUI (2) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (15-Jul-2019, 2:23:43 am)

```
passing class name :- Example.java
failing class name :- Example1.java
============================================ passing class analysis ============================================

+-----------------------+-----------------------+-----------------------+
|Method                 |                   Time|              Frequency|
+-----------------------+-----------------------+-----------------------+
|calculateMessage       |   8.583949999945162E-4|                      1|
|calculatediv           |   4.744999998251842E-6|                      1|
|main                   |    0.00990182500000003|                      1|
|calculateSum           |   8.613139999980035E-4|                      1|
+-----------------------+-----------------------+-----------------------+

first class time duration :- 0.00990182500000003



================================Exception exists================================
exception found in method :- calculateMessage()  (Example1.java:5)



============================================ failing class analysis ============================================

+-----------------------+-----------------------+-----------------------+
|Method                 |                   Time|              Frequency|
+-----------------------+-----------------------+-----------------------+
|calculateMessage       |   1.831450000011614E-4|                      1|
|main                   |  1.9700800000066465E-4|                      1|
|calculateSum           |  1.8606400000464873E-4|                      1|
+-----------------------+-----------------------+-----------------------+
```

OUPUT ON CONSOLE

```
MethodTraceGUI (2) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (15-Jul-2019, 2:23:43 am)
|Method                  |                     Time|                Frequency|
+------------------------+-------------------------+-------------------------+
|calculateMessage        |    8.583949999945162E-4|                        1|
|calculatediv            |    4.744999998251842E-6|                        1|
|main                    |     0.00990182500000003|                        1|
|calculateSum            |    8.613139999980035E-4|                        1|
+------------------------+-------------------------+-------------------------+

first class time duration :- 0.00990182500000003



=================================Exception exists================================
exception found in method :- calculateMessage()  (Example1.java:5)



============================================= failing class analysis =============================================

+------------------------+-------------------------+-------------------------+
|Method                  |                     Time|                Frequency|
+------------------------+-------------------------+-------------------------+
|calculateMessage        |    1.831450000011614E-4|                        1|
|main                    |   1.9700800000066465E-4|                        1|
|calculateSum            |   1.8606400000464873E-4|                        1|
+------------------------+-------------------------+-------------------------+

====================  time differenece b/w same methods of passing case and failing case =====================================
*) calculateMessage ===>6.752499999933548E-4
*) main ===>0.009704816999999366
*) calculateSum ===>6.752499999933548E-4
```

GRAPHICAL OUPUT OF METHODS DURATION



WORKING

GUI :-

Developed GUI using window builder in eclipse
It consist of two text fields for writing both passing and failing java class names and a compare button which on clicked will do following tasks:-

- Print name of both class on console
- Call classes which contains functionality of generating log files , comparing them and find the exceptions
- , Find number of times each method called and time taken by it , calculate difference in execution of each method for both the classes
- The output is shown on the console and a gui window for graph

WORKING

Implementation:-

- First the user have to provide names of both files he wants to compare and then click on submit button
- After that using filel/O in java application will write command of execution which is of form "javac file_name.java" in commands.txt file
- And using processBuilder class it will generate the output and error file (if exist) in output.txt and error.txt
- Then it will execute commands written in another command.txt file which consist of commands for generating trace files with .trc extension
- Then it will generate human readable or log file using trace files generated in previous steps

WORKING

Implementation:-

- After generating log files it will compare them to find :-

  1) whether exception exist or not :- it can be solved by parsing log files and comparing the codeflow of execution in them

  2) if it exist find it's location:- it can be find using jstacktrace of log files
  3) how much time each method is consuming:- using time stamp of entry and exist of each method
  4) number of times each method is called:- using codeflow count each time when a method is getting entry in entry
  5) time difference between same methods in different cases:- using stored results of problem 3's solution
  6) time difference between both class to execute

WORKING

Implementation:-

- Ouptut for all the operations performed will be shown on console that can be done pretty easily using java
- And a graphical representation of methods using java library JFreeChart
- And for generating tables it will use wagu package which is available on github

For more explanation watch video available on github :-

https://github.com/jatinjha/method-trace-analyzer-ibm.git

THANK YOU

FOR ANY QUERIES

CONTACT :-

JATINJHA370@GMAIL.COM