# **Team Name: Brainwave**

# **METHOD TRACE ANALYZER**

Java based GUI application to parse and compare multiple trace files, to help debugging the code. Method trace is used for post-mortem debugging. It consists of timestamp of entry and exit points and also execution time for each method invocation. They may also contain stack-trace for each invocation. In case of functionality issues, comparing of the data is used for debugging.

**Functionalities included**

1. Compare two trace files: one from failing and passing case each and find out the anomaly.
2. Calculate the execution time of each method and compare the extra time taken. This will be helpful in addressing hang and performance related problems.
3. Parse one or more trace files and create a tabular view for the number of times each method is invoked.
4. Compare code-flow and stack trace for failing and passing case and find anomalies.

**Technology Stack**

1. Java
2. IBM Java Method trace
3. Eclipse Photon
4. Window Builder
5. Open JDK with OpenJ9

**Description of files uploaded**

1. IbmGuiWindow.java - Includes java code for GUI
2. LogAnalysis.java - Includes java code for implementation of Method-Trace Analyzer
3. SyncPipe.java - Supporting doc for running Xtrace commands in Command Prompt through Java
4. ppt.pdf - Presentation based on idea
5. undertaking.pdf - Documentation of Undertaking
6. video.mp4 - Video showing implementation of functionalities
7. Abcd.java- passing java file
8. Abcdex.java-failing java file with exception
9. AbcdTrace.trc- trace file for passing class
10. AbcdexTrace2.trc- trace file for failing class
11. AbcdlogFile.log- log file for Abcd class file
12. AbcdexlogFIle2.log- log file for Abcdex

**Team Members Details**

1.Tanmay Shriwas

2.Roopali Mishra

3.Simranjeetsingh Saluja

**Team Members Roles**

1. Tanmay shriwas:- Development, Xtrace, log generation,loganalysis.
2. Roopali Mishra:- debugging, gui(window builder),testing.
3. Simranjeetsingh saluja:- Searching and testing.

**Workflow Of The Entire Project**

**Setting up all software and technologies used in the project**

1. Open J9 VM and Xtrace

OpenJ9 VM tracing is a powerful feature to help you diagnose problems with minimal affect on performance. Tracing is enabled by default, together with a small set of trace points going to memory buffers. You can enable tracepoints at run time by using levels, components, group names, or individual tracepoint identifiers to trace VM internal operations and instrumented Java™ applications. You can also trace Java methods.

Trace data can be output in human-readable or in compressed binary formats. The VM provides a tool to process and convert the compressed binary data into a readable format.

1. Trace Formatter

The trace formatter is a Java™ program that converts binary trace point data in a trace file to a readable form. The formatter requires the TraceFormat.dat and J9TraceFormat.dat files, which contain the formatting templates. The formatter produces a file that contains header information about the VM that produced the binary trace file, a list of threads for which trace points were produced, and the formatted trace points with their time stamp, thread ID, trace point ID, and trace point data.

1. OpenJDK with OpenJ9

The community develop and maintain a build and test infrastructure for the OpenJDK source across a broad range of platforms. For information about the platforms and minimum operating system levels supported for the builds, see the AdoptOpenJDK

1. Eclipse Photon

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment.

1. Window Builder

SWT Designer is a visual editor used to create graphical user interfaces. It is a two way parser, e.g., you can edit the source code or use a graphical editor to modify the user interface. SWT Designer synchronizes between both representations. SWT Designer is part of the WindowBuilder project

**The flow of the project: how it works**

1. Firstly, the user needs to enter two java files in the given text fields. These should be the ones specifying the passing and failing case of each program.
2. Press the submit button.
3. This will lead to execution of the program in the background.
4. The file names entered will be captured as strings in the code.
5. Then, the Xtrace commands will generate trace files(.trc) for the entered files.
6. After that, the command to generate the log files from the trace files will be implemented.
7. Log files of the passing and failing case are compared to determine what anomalies(exceptions) led to the failing of the program.
8. Stack trace is compared to determine when and which line from the code leads to the exception.
9. Time taken by each method in the class file to execute is shown, thus helping to know which method took more time.
10. And lastly it will also give the count of number of times each method of the program is invoked.
11. Output is then displayed on the cmd.