Ideation Document

Name of the problem statement:

Problem statement- 2: Method-trace Analyser

Problem Statement-2

Method Trace Analyse is a real-life based problem which aims to make the work of understanding method trace log easier and less time consuming. These method trace logs can contain time stamp, event type, event summary, jStacktrace and much more.

Uniqueness/features added to the problem statement

In order to assist developers in comparing log files and as well as to deal with multiple failing logs with respect to passing logs. We decided to come up with 2 certain features namely, “Batch Anomalies” and “One to many Anomalies”. In Batch Anomalies, all the log files whether they are failing or passing logs are compared with one another and is then compared to address issues like performance/hang issues, longer execution time, methods which are not completing its execution, etc. In One to Many Anomalies, one log file which is considered as passing log file is compared with other failing logs and detailed report of each issues like different codeFlow, jStacktrace and Exceptions is shown in accordion view with tabs.

Impact on Businesses

Since this problem statement targets developers working on Java projects, it will save a lot of time by instead of reading logs of multiple files and finding issues than the work automated by Method Trace Analyzer. It will directly boast the process flow of fixing bugs and developing application in businesses.

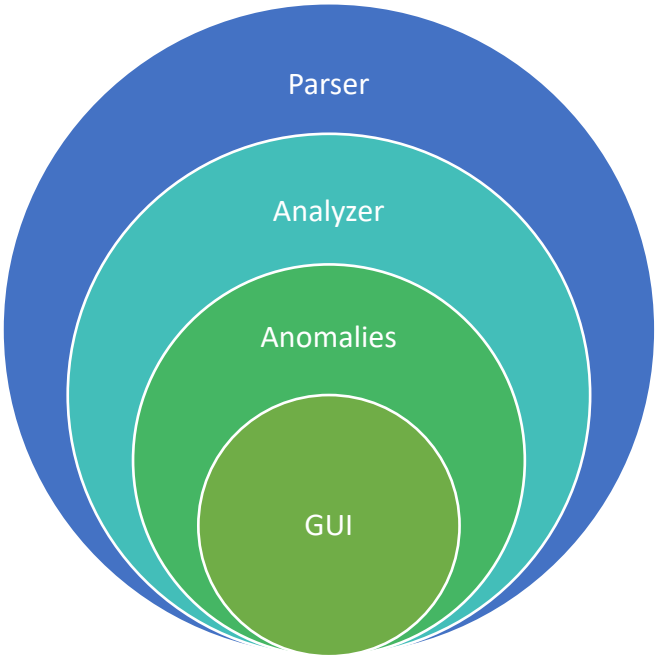
Scope of work

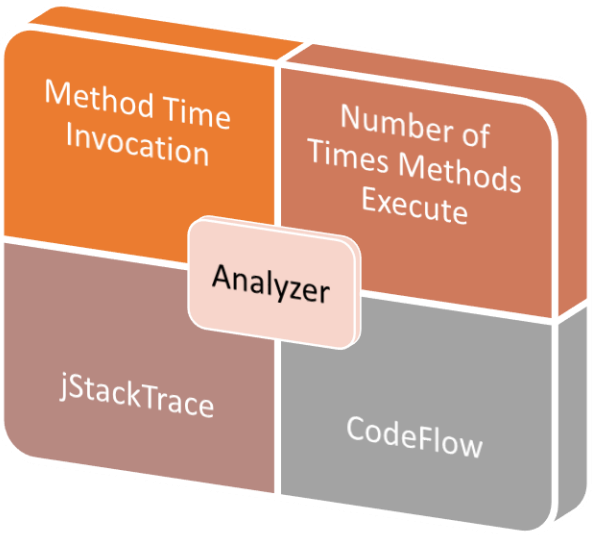
Method Trace Analyser is very useful in case of tracing huge data, the application can be embedded in IBM library and developers can take advantage of its efficiency and time factor. In this way, debugging code will fully become fully automated and can be used for big java applications.

The developed solution has 4 modules: -

1. Anomalies between multiple files (Batch Anomalies) and One to Many Anomalies which compares a passing log with multiple failing logs.
2. CodeFlow (which shows the sequence of the methods during runtime)
3. Method time invocation for each method in milliseconds
4. Number of times each method executed.
5. jStackTrace (for getting java stack trace from the specific error or function)

Architectural Flow





Technologies/Platform/APIs to be used

- OpenJDK 8
- NetBeans IDE 8.2
- Apache Maven 3.39
- JAVAFX with FXML for GUI rendering
- Apache Commons IO for reading File to String

Team Details:

Team Name: JavaMonks

Team Size: 1

Team Members:

1. Member 1: Manjot Singh Sidhu

   Email: manjot.singh@btech.christuniversity.in

Role of Each Team Member

Manjot Singh Sidhu

- Solution Architecture, Design and Development
- Project Lead