Git Interactive Tool

# Overview:

**G**it **I**nteractive **T**ool is a Java-based command-line tool that allows users to query the git repositories to capture different metrics.

The tool currently supports metrics viz., topNStars, topNForks, topNPullRequest, topNContribution and interfaces with the Github V3 APIs. Git Interactive Tool uses Gradle to manage dependencies and building artifact, Testng framework for unit tests, Apache CLI library for command-line options.

# Usage guide for Git Interactive Tool:

|  |  |  |
| --- | --- | --- |
| Command line option | Takes args | Description |
| -l | yes | (Optional option) Set limit of how many repo to search to build the result.  If this is not set then all the repos of github will be scanned. |
| -m | yes | (Compulsory option) Enter the operation  (Supported operations topNStars, topNForks, topNPullRequest, topNContribution). |
| -n | yes | (Compulsory option) Show results for these top n repos. |
| -s,--sleep-for-rate | no | (Optional option) Just sit silently till new API requests are available. |
| -t | yes | (Optional option) OAuth2 token - to increase request count to 5000 request per hour. |
| -u,--usage | no | Shows usage information for this tool. |

# Source code:

<https://github.com/iyervijaywork/GitInteractiveTool>

# How to Build:

$ cd <repo\_dir>/GitInteractiveTool

$ ./gradlew assemble

$ ./gradlew build

# Commands Examples:

|  |  |
| --- | --- |
| Command | Description |
| $ java -jar build/libs/GitInteractiveTool.jar --usage | Print usage using the following command |
| $ java -jar build/libs/GitInteractiveTool.jar -n 2 -m topNStars | Print top 2 github repository with highest stars |
| $ java -jar build/libs/GitInteractiveTool.jar -n 2 -m topNForks | Print top 2 github repository with highest forks |
| $ java -jar build/libs/GitInteractiveTool.jar -n 2 -m topNPullRequest | Print top 2 github repository with highest pull requests |
| $ java -jar build/libs/GitInteractiveTool.jar -n 2 -m topNContribution | Print top 2 github repository with highest contribution |
| $ java -jar build/libs/GitInteractiveTool.jar -n 2 -m topNContribution --sleep-for-rate | Print top 2 github repository with highest contribution. To avoid having tool exiting when the rate limit is exceeded, we can use the --sleep-for-rate option. When this option is used, instead of exiting, when the rate timit is reached tool will just sit silently, waiting until new API requests are available: |
| $java -jar build/libs/GitInteractiveTool.jar -n 2 -m topNContribution --sleep-for-rate -t <Personal\_access\_tokens> | Print top 2 github repository with highest contribution. To avoid having tool exiting when the rate limit is exceeded, we can use the --sleep-for-rate option. When this option is used, instead of exiting, when the rate timit is reached tool will just sit silently, waiting until new API requests are available. -t option allows you to pass your personal access token (OAuth2 token) which will increase your api rate limit to 5000 request per hour |
| $ java -jar build/libs/GitInteractiveTool.jar -n 2 -m topNContribution --sleep-for-rate -l 30 | Print top 2 github repository with highest contribution but only scan the first 30 repository. |
| $ java -Xdebug -Xrunjdwp:transport=dt\_socket,address=8000,server=y,suspend=y -jar build/libs/GitInteractiveTool.jar -n 2 -m topNStars | Print top 2 github repository with highest stars with debug option to debug tool with source code. |

# Testing

1. ***Automated tests:***

|  |  |
| --- | --- |
| Test method | Description |
| ProgramTest.processRequestTest | Test the processRequest API for different n and mode values.  This testNG test uses data providers to test for different values of n and modes. |
| RepoMetricComparator.compareTest1 | Test the repo metric comparator a < b |
| RepoMetricComparator.compareTest2 | Test the repo metric comparator a > b |
| UtilsTest.topNForksTest | Test the API execution engine in Utils with TopNForks url strings |
| UtilsTest.topNStarsTest | Test the API execution engine in Utils with TopNStars url strings |

1. ***Manual tests:***

|  |  |
| --- | --- |
| Test | Description |
| Run tool in different valid modes with the following values of n  -1, 0, 2, 10, 1000 | With -1 and 0 value of N the tool should throw correct exception handling message. For the rest of the scenarios verify correct output. Verify that metric count is currectly sorted in descendin order |
| Run tool with n = 2 but an invalid mode. Use alphanumeric values including special characters. | the tool should throw correct exception handling message for invalid/unsupported modes. |
| Run the tool to find topN pull requests limiting search to only first 30 git repositories. | verify correct output. Verify TopN is picked for only first 30 repos. |
| Run the tool to find topN contribution limiting search to only first 30 git repositories. | verify correct output. Verify TopN is picked for only first 30 repos. Verify contribution is computed correctly as (total number of pull request)/(total forks) |
| Run the tool without a personal access token and with --sleep-for-rate option | Verify output. Verify that since without authentication the rate is only 60 requests per hour, the tool correctly sleeps and polls for new API request rate to become available before continuing further. |
| Run tool without passing any options | Verify that the tool correctly prompts for mandatory options to be passed. |
| Run tool with unsupported options | the tool should throw correct exception handling message for invalid/unsupported options. |