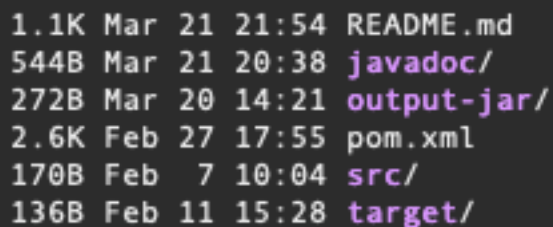


GetIssues version 1.0
Mark Morris

I developed this application on my MacBook Pro with Java 8. So, you need Java 8 installed. I tested on Windows 8.1 and it was really really slow because of the security software running and consuming the majority of the cpu while inspecting every byte flowing to the application. So keep this in mind. When you see 2 minutes compared 2.4 seconds.

Below are the directories and files that make up the application I developed for this exercise. I'll explain each one:



```
1.1K Mar 21 21:54 README.md
544B Mar 21 20:38 javadoc/
272B Mar 20 14:21 output-jar/
2.6K Feb 27 17:55 pom.xml
170B Feb 7 10:04 src/
136B Feb 11 15:28 target/
```

The directory called javadoc contains the generated java documentation rendered in HTML.

The directory called output-jar contains work files for easy execution of the application. I'll discuss it below.

The pom.xml is the maven file for the project. I didn't use it, because I used:

Spring Tool Suite 4

Version: 4.1.0.RELEASE
Build Id: 201812201400

Copyright (c) 2007 - 2018 Pivotal, Inc.

to develop the application and exercised the IDE's tools to develop, test, troubleshoot, generate, and document the application. The pom.xml will build (compile and package into a jar).

And "drum roll"...the src directory is the code.

So, let me describe the output-jar directory. It serves as my launch

pad. I use my IDE to generate a "runnable" jar and output it to that directory. Then I open a terminal and execute it.

This let's me code and test in the IDE and when I reach a milestone I generate the runnable jar and see how it performs from the user's perspective, in this case the command line. This also let's the end user execute it immediately provided they have a working Java 8 runtime environment.

Here is the output-jar directory:

```
10K Mar 21 21:15 build.xml
11M Mar 21 21:15 getIssues.jar
102B Mar 21 19:48 jar/
7.5K Mar 21 21:15 jar-in-jar-loader.zip
1.8K Mar 21 19:38 setclasspath
```

The first file called build.xml is an Ant file that "on my machine" will build the "runnable" jar from the command line. It works for me. How many times have you heard that :) It uses the jar-in-jar-loader.zip file. Both of these files are generated from the IDE as an option when generating the runnable jar.

The file called getIssues.jar is the "runnable" jar. You can execute the application by entering: `java -jar getIssues.jar` and it will execute and output some help informing the user on it's execution parameters and some notes on performance and utility.

The directory called jar is where I copy the "runnable" jar to and expand it with the command: `jar xf getIssues.jar`

The jar xf command is like unzip and will unjar the contents, which provides the ability to examine the contents and execute the application from outside the jar, which improves performance. I do this to compare the performance between the two to get an idea of the performance penalty for the ease of execution.

When doing this you need a classpath to be able to resolve the references, which can be a chore setting up correctly. So, I have provided the file called setclasspath to do this for you.

What you need to do is edit this simple text file and then perform the source command. I do this from the jar directory, so I enter: `source ../setclasspath` and it sets the environment for execution. It is easy to edit and source, and then execute the application like

this, from the jar directory: `java mm/issues/Main`

You will see the same output as you saw from the runnable jar.

Here are the steps described above:

```
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar
[Thu Mar 21 @ 10:52 markmorris$ ll
total 22328
-rw-r--r-- 1 markmorris wheel 10K Mar 21 21:15 build.xml
-rw-r--r-- 1 markmorris wheel 11M Mar 21 21:15 getIssues.jar
drwxr-xr-x 3 markmorris wheel 102B Mar 21 19:48 jar/
-rw-r--r-- 1 markmorris wheel 7.5K Mar 21 21:15 jar-in-jar-loader.zip
-rw-r--r--@ 1 markmorris wheel 1.8K Mar 21 19:38 setclasspath
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar
[Thu Mar 21 @ 10:52 markmorris$ cd jar
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
[Thu Mar 21 @ 10:52 markmorris$ cp ../getIssues.jar .
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
[Thu Mar 21 @ 10:52 markmorris$ jar xf getIssues.jar
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
[Thu Mar 21 @ 10:53 markmorris$ source ../setclasspath
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
[Thu Mar 21 @ 10:53 markmorris$ java mm/issues/Main
**** NO INPUT ARGUMENTS FOUND. APPLICATION REQUIRES INPUT ARGUMENT ****
getIssues version 1.0
Usage: java -jar getIssues.jar [p] owner1/repo1 owner2/repo-2 owner3/repo_3 ...
The optional first argument p instructs the code to return the JSON string in pretty format.
NOTE: GitHub enforces a rate limit of 10 per minute and you will experience it when you enter
many owner/repo entries or you continuously execute too fast.
You can login to GitHub by entering: your_username and your_password
to improve your rate limit. The rate limit is increased to 30.
Set the following environment variables to login to github.
GITHUB_ISSUES_USERNAME=username
GITHUB_ISSUES_PASSWORD=password
Set this environment variable if you want to print job stats at the end.
GITHUB_ISSUES_PRINT_MESSAGES=true
Set this environment variable if you want to ensure you get all
issues available and avoid the rate limit.
This will cause the program to sleep for 30 seconds when it
detects a potential rate limit violation approaching.
GITHUB_ISSUES_PACING=true

~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
[Thu Mar 21 @ 10:54 markmorris$ ]
```

Example commands:

Note: All of these commands apply to both methods of execution.

```
java -jar getIssues.jar tarantulatechnology/apache
```

```
java -jar getIssues.jar tarantulatechnology/apache tarantulatechnology/
bitcoin
```

You can add as many repositories as you like. But rate limits apply as noted in the application output in the screen shot above.

Remember, you can edit the `setclasspath` file and enable helpful printing and or application messages to understand execution.

You can fetch 1000's of issues regardless of the rate limit constraints imposed by GitHub--10 a minute and 30 a minute--if you set `pacing` true.

To get a rate limit of 30 a minute, instead of the default 10 a minute, set the environment settings for `username` and `password` to inform the application you want authenticated requests. There is be a slight performance penalty, but it will not effect the overall performance.

To print out JSON--you can actually read--use this little puppy :) the lowercase letter `p` which MUST be the first command argument, if you want JSON pretty print.

Ex. `java -jar getIssues.jar p tarantulatechnology/apache`

Notes on pacing:

You can use `pacing` to defeat the rate limit from causing you not get all of the issues. What happens is if you breach the rate limit, GitHub will send back a message indicating this state. When this happens you must wait a period of time to allow the GitHubs rate limiting clock to reset. When it does reset you can continue fetching issues. If you run without `pacing` enabled you will most likely not be able to get all of the issues, and what you get will be incomplete and not useful, so use `pacing` to ensure you get them all. This is not a free lunch! You will see the application sleep for 30 seconds multiple times combating the rate limit. But you will get the integrity of data you demand, and it is not that bad. I was able fetch 20,000 issues in ~25 minutes :) Not one was missed! And it ran unattended to completion.

What else...

Oh, I did some testing, but need to create a nice suite of unit tests that execute the helpers and perform requests. I did document what came off the top of my head as to what I would consider creating unit tests for.

This was a fun sprint and code is in a place where it can meet immediate utility, and can evolve to meet the additional qualities that are required to ensure a minimum of technical debt.

I hope you enjoy playing with the application as much as I enjoyed creating it :)

Sample commands and output:

```
java mm/issues/Main tarantulatechnology/apache
```

```
[Thu Mar 21 @ 11:24 markmorris$ java mm/issues/Main tarantulatechnology/apache
{"issues":[{"id":229659214,"state":"open","title":"Fix broken url to \"Better Bloom Filter\" paper","reposito
ry":"TarantulaTechnology/apache","created_at":"2017-05-18T12:47:31Z"}, {"id":406961191,"state":"closed","title
":"My Test Issues","repository":"TarantulaTechnology/apache","created_at":"2019-02-05T20:29:40Z"}], "top_day":
{"day": "2019-02-05", "occurrences": {"TarantulaTechnology/apache": 1, "tarantulatechnology/apache": 0}}}
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
```

```
java mm/issues/Main p tarantulatechnology/apache
```

```
[Thu Mar 21 @ 11:24 markmorris$ java mm/issues/Main p tarantulatechnology/apache]

{
  "issues":[
    {
      "id":229659214,
      "state":"open",
      "title":"Fix broken url to \"Better Bloom Filter\" paper",
      "repository":"TarantulaTechnology/apache",
      "created_at":"2017-05-18T12:47:31Z"
    },
    {
      "id":406961191,
      "state":"closed",
      "title":"My Test Issues",
      "repository":"TarantulaTechnology/apache",
      "created_at":"2019-02-05T20:29:40Z"
    }
  ],
  "top_day":{
    "day":"2019-02-05",
    "occurrences":{
      "TarantulaTechnology/apache":1,
      "tarantulatechnology/apache":0
    }
  }
}
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
Thu Mar 21 @ 11:26 markmorris$
```

Use the "time" command for system use: `time java mm/issues/Main p tarantulatechnology/apache`

```
[Thu Mar 21 @ 11:26 markmorris$ time java mm/issues/Main p tarantulatechnology/apache

{
  "issues":[
    {
      "id":229659214,
      "state":"open",
      "title":"Fix broken url to \"Better Bloom Filter\" paper",
      "repository":"TarantulaTechnology/apache",
      "created_at":"2017-05-18T12:47:31Z"
    },
    {
      "id":406961191,
      "state":"closed",
      "title":"My Test Issues",
      "repository":"TarantulaTechnology/apache",
      "created_at":"2019-02-05T20:29:40Z"
    }
  ],
  "top_day":{
    "day":"2019-02-05",
    "occurrences":{
      "TarantulaTechnology/apache":1,
      "tarantulatechnology/apache":0
    }
  }
}

real    0m2.090s
user    0m2.266s
sys      0m0.186s
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
Thu Mar 21 @ 11:28 markmorris$
```

With environment variables for pacing and messages set to true:

```
setclasspi ●
```

```
1  #!/bin/bash
2
3  #classpath for mm.issues.Main
4  export CLASSPATH=./:./antlr-runtime-3.5.2.jar:./aop
5  • 1.0.4.jar:./javax.json-api-1.0.jar:./javax.ws.rs-ap
6  • org.junit.jupiter.api_5.3.1.v20181005-1442.jar:./or
7  • org.junit.vintage.engine_5.3.1.v20181005-1442.jar:
8
9  export GITHUB_ISSUES_USERNAME=
10 export GITHUB_ISSUES_PASSWORD=
11
12 export GITHUB_ISSUES_PRINT_MESSAGES=true
13
14 export GITHUB_ISSUES_PACING=true
```

```
time java mm/issues/Main p tarantulatechnology/apache
```



```

[Thu Mar 21 @ 11:31 markmorris$ time java mm/issues/Main p tarantulatechnology/apache
ratelimit is [10]
rateLimitRemaining is [8]

{
  "issues":[
    {
      "id":229659214,
      "state":"open",
      "title":"Fix broken url to \"Better Bloom Filter\" paper",
      "repository":"TarantulaTechnology/apache",
      "created_at":"2017-05-18T12:47:31Z"
    },
    {
      "id":406961191,
      "state":"closed",
      "title":"My Test Issues",
      "repository":"TarantulaTechnology/apache",
      "created_at":"2019-02-05T20:29:40Z"
    }
  ],
  "top_day":{
    "day":"2019-02-05",
    "occurrences":{
      "TarantulaTechnology/apache":1,
      "tarantulatechnology/apache":0
    }
  }
}

Util.validateInputArguments: 1 valid repositories.

GitHubIssueProvider.searchForIssues: Last page for tarantulatechnology/apache is 1

CommandLineProcessor.run: Time executing AsyncConcurrentSearch PT1.468S

Builder.buildIssues: Count for TarantulaTechnology/apache issues: 2
Builder.buildIssues: Count for all issues collected: 2
Builder.buildIssuesList: Count for issues: 2

JSON Report Length 433

Time executing CommandLineProcessor PT1.539S

real    0m1.760s
user    0m2.265s
sys     0m0.184s
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
Thu Mar 21 @ 11:32 markmorris$

```

With no authentication and pacing set to true -- you only have 10/min and this code is concurrent and async, so it will use all the cores. I could have used an executor and limited the threads, but for another day :)

```
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
[Thu Mar 21 @ 11:47 markmorris$ time java mm/issues/Main p bitcoin/bitcoin
ratelimit is [10]
rateLimitRemaining is [8]
ratelimit is [10]
rateLimitRemaining is [7]
ratelimit is [10]
rateLimitRemaining is [6]
ratelimit is [10]
rateLimitRemaining is [5]
ratelimit is [10]
rateLimitRemaining is [4]
Rate limit is close to max, so sleeping for 30 seconds.
```

Authentication increases the rate limit, and the code increases the gate to 10. I can improve this a lot, but it serves the purpose for now.

```
~/Documents/workspace-spring-tool-suite-4-4.0.0.RELEASE-Github/markmorris-github-issues/output-jar/jar
[Thu Mar 21 @ 11:50 markmorris$ time java mm/issues/Main p bitcoin/bitcoin
ratelimit is [30]
rateLimitRemaining is [29]
ratelimit is [30]
rateLimitRemaining is [28]
ratelimit is [30]
rateLimitRemaining is [27]
ratelimit is [30]
rateLimitRemaining is [26]
ratelimit is [30]
rateLimitRemaining is [25]
ratelimit is [30]
rateLimitRemaining is [24]
ratelimit is [30]
rateLimitRemaining is [23]
ratelimit is [30]
rateLimitRemaining is [22]
ratelimit is [30]
rateLimitRemaining is [21]
ratelimit is [30]
rateLimitRemaining is [20]
ratelimit is [30]
rateLimitRemaining is [19]
ratelimit is [30]
rateLimitRemaining is [18]
ratelimit is [30]
rateLimitRemaining is [17]
ratelimit is [30]
rateLimitRemaining is [16]
ratelimit is [30]
rateLimitRemaining is [15]
ratelimit is [30]
rateLimitRemaining is [14]
ratelimit is [30]
rateLimitRemaining is [13]
ratelimit is [30]
rateLimitRemaining is [12]
ratelimit is [30]
rateLimitRemaining is [11]
ratelimit is [30]
rateLimitRemaining is [10]
ratelimit is [30]
rateLimitRemaining is [9]
Rate limit is close to max, so sleeping for 30 seconds.
```

Okay. That's ALL FOLKS :)

-Mark