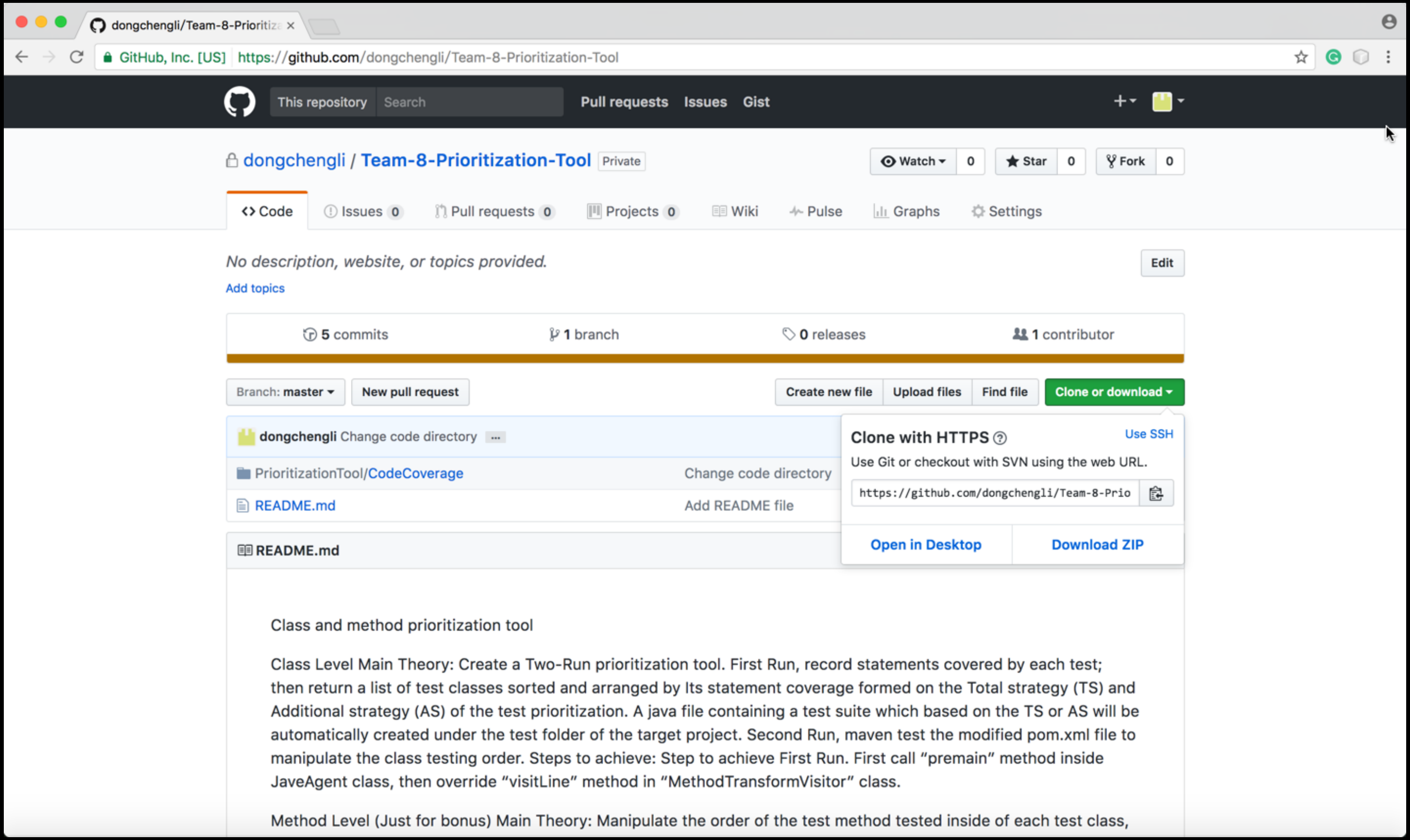
Download Prioritization Tool from the GitHub repository, https://github.com/dongchengli/Team-8-Prioritization-Tool



No description, website, or topics provided.

dongchengli Change code directory

PrioritizationTool/CodeCoverage	Change code directory
README.md	Add README file

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

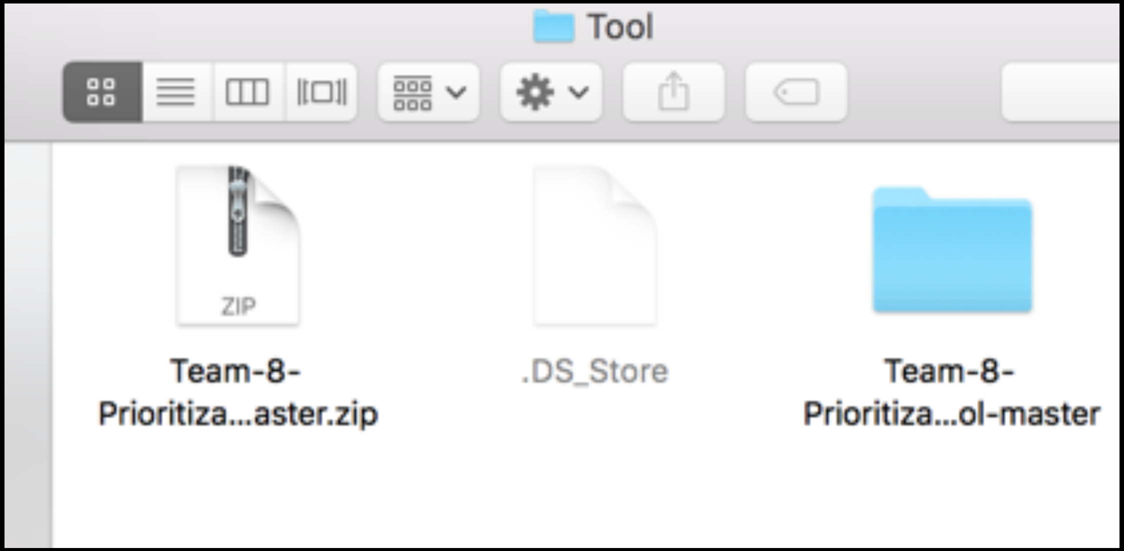
`https://github.com/dongchengli/Team-8-Prio`

README.md

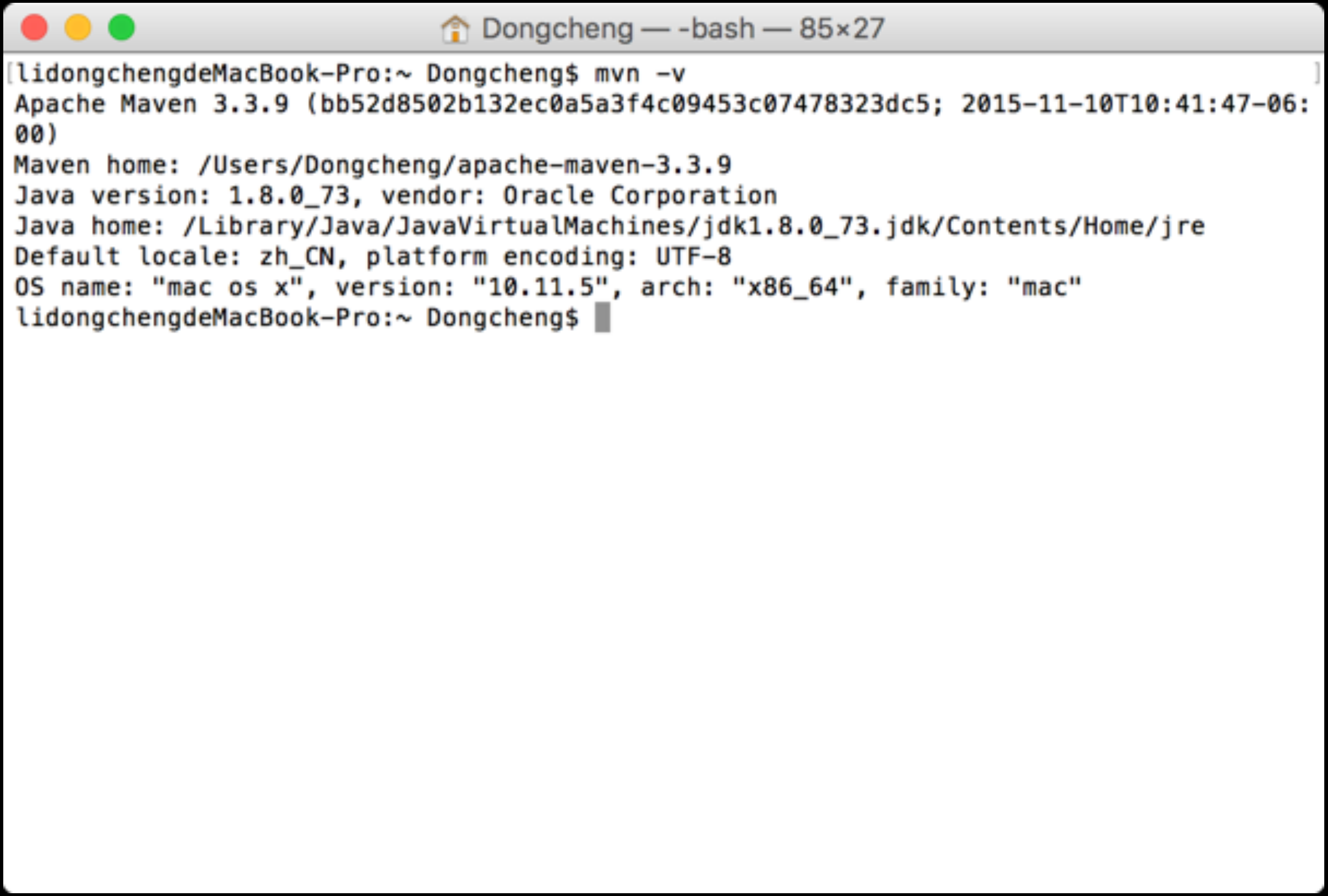
Class and method prioritization tool

Class Level Main Theory: Create a Two-Run prioritization tool. First Run, record statements covered by each test; then return a list of test classes sorted and arranged by Its statement coverage formed on the Total strategy (TS) and Additional strategy (AS) of the test prioritization. A java file containing a test suite which based on the TS or AS will be automatically created under the test folder of the target project. Second Run, maven test the modified pom.xml file to manipulate the class testing order. Steps to achieve: Step to achieve First Run. First call “premain” method inside JaveAgent class, then override “visitLine” method in “MethodTransformVisitor” class.

Method Level (Just for bonus) Main Theory: Manipulate the order of the test method tested inside of each test class, Put Downloaded Team-8-Prioritization-Tool-master.zip file inside a new directory or any directory (Directory named simple and with no space between words is recommended, such as Tool), and extract the zip file.

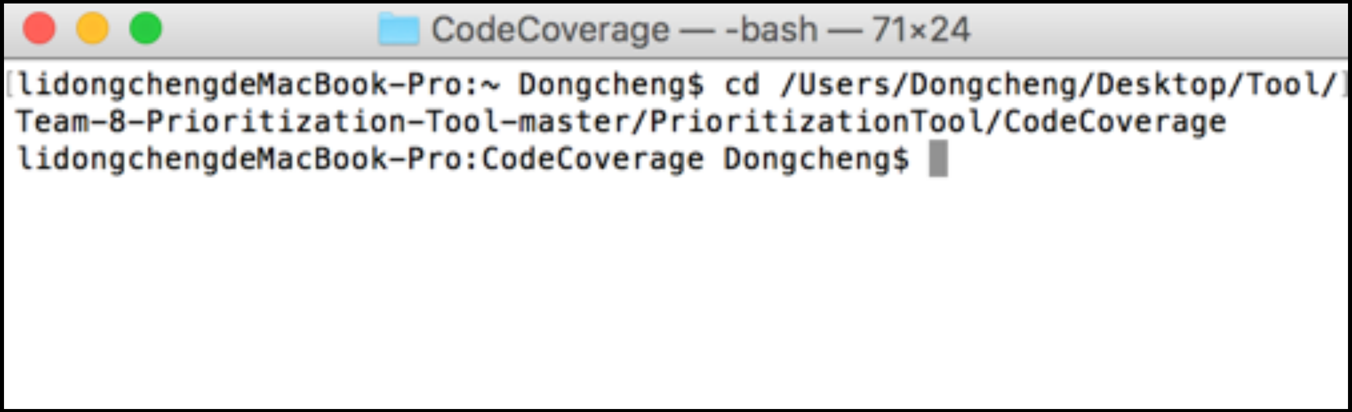


From the commend line, first check the maven, type mvn –v, if no maven installed, it need to be installed



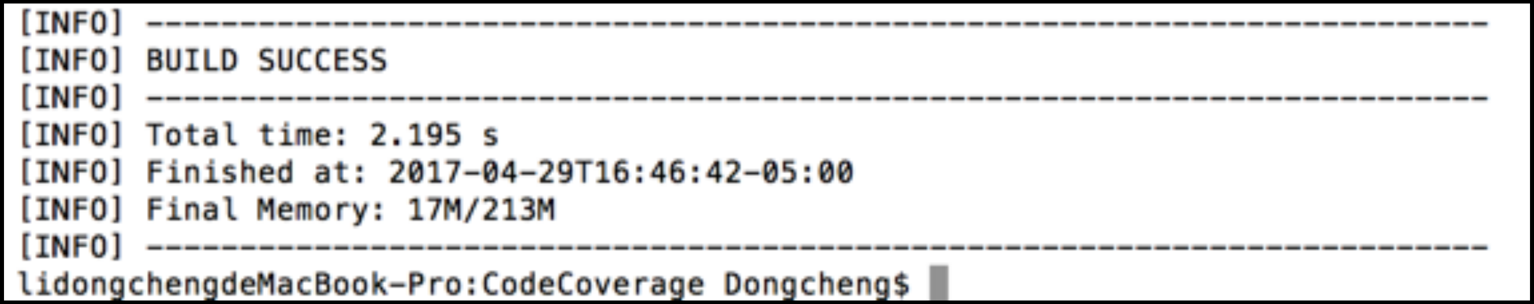
```
lidongchengdeMacBook-Pro:~ Dongcheng$ mvn -v
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T10:41:47-06:
00)
Maven home: /Users/Dongcheng/apache-maven-3.3.9
Java version: 1.8.0_73, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/Contents/Home/jre
Default locale: zh_CN, platform encoding: UTF-8
OS name: "mac os x", version: "10.11.5", arch: "x86_64", family: "mac"
lidongchengdeMacBook-Pro:~ Dongcheng$ 
```

In the Commend Line Direct to the directory of the CodeCoverage folder inside of the Team-8-Prioritization-Tool-master folder, type cd /yourfolderpath/



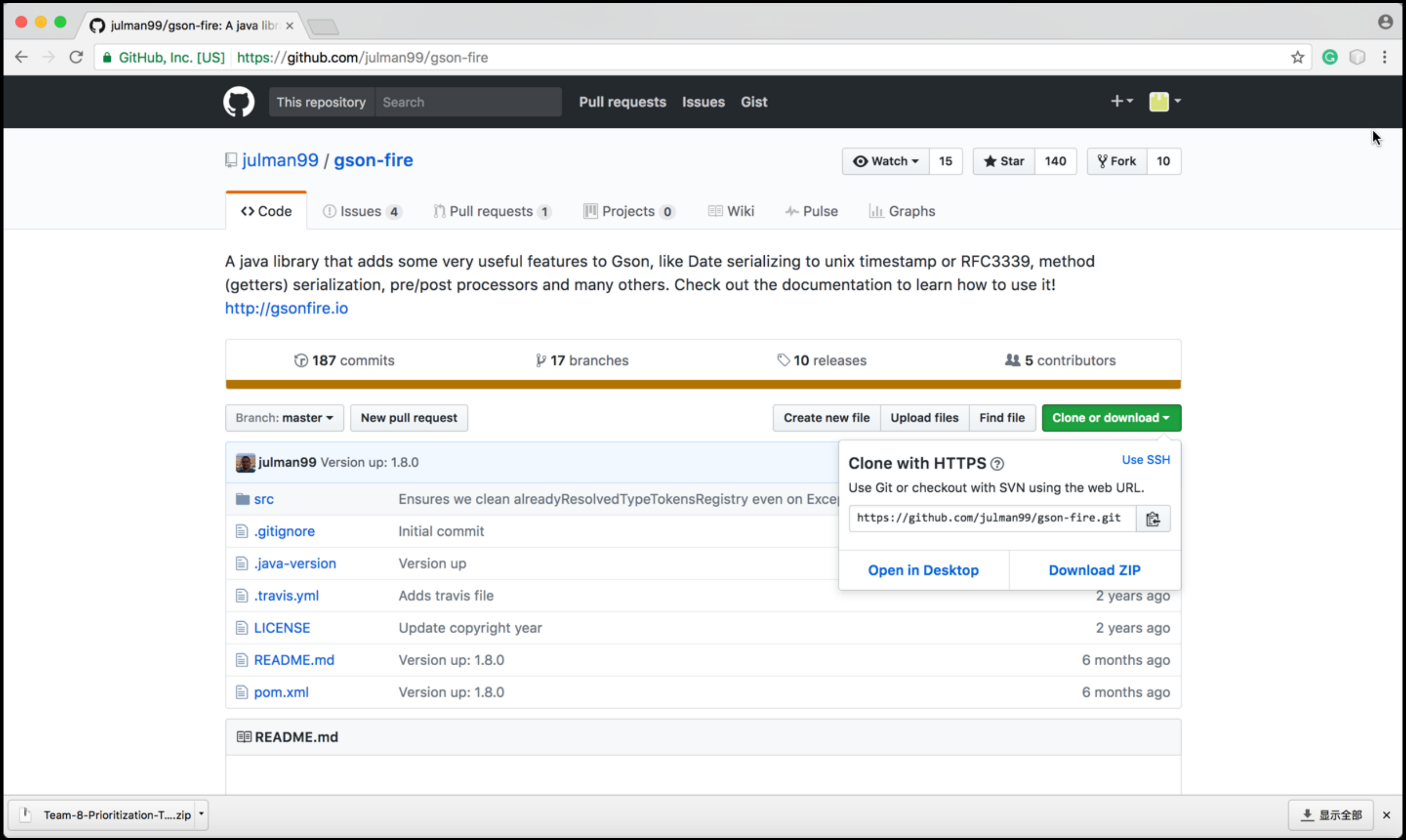
```
lidongchengdeMacBook-Pro:~ Dongcheng$ cd /Users/Dongcheng/Desktop/Tool/
Team-8-Prioritization-Tool-master/PrioritizationTool/CodeCoverage
lidongchengdeMacBook-Pro:CodeCoverage Dongcheng$
```

Then press mvn install, and it will show build successfully



```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.195 s
[INFO] Finished at: 2017-04-29T16:46:42-05:00
[INFO] Final Memory: 17M/213M
[INFO] ------------------------------------------------------------------------
lidongchengdeMacBook-Pro:CodeCoverage Dongcheng$
```

Go to https://github.com/julman99/gson-fire, then download this real world Github project for testing purpose.


julman99/gson-fire: A java libr

GitHub, Inc. [US] https://github.com/julman99/gson-fire

This repository Search Pull requests Issues Gist

julman99 / gson-fire

Watch 15 Star 140 Fork 10

Code Issues 4 Pull requests 1 Projects 0 Wiki Pulse Graphs

A java library that adds some very useful features to Gson, like Date serializing to unix timestamp or RFC3339, method (getters) serialization, pre/post processors and many others. Check out the documentation to learn how to use it! http://gsonfire.io

187 commits 17 branches 10 releases 5 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

julman99 Version up: 1.8.0

src	Ensures we clean alreadyResolvedTypeTokensRegistry even on Excep	
.gitignore	Initial commit	
.java-version	Version up	
.travis.yml	Adds travis file	2 years ago
LICENSE	Update copyright year	2 years ago
README.md	Version up: 1.8.0	6 months ago
pom.xml	Version up: 1.8.0	6 months ago

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

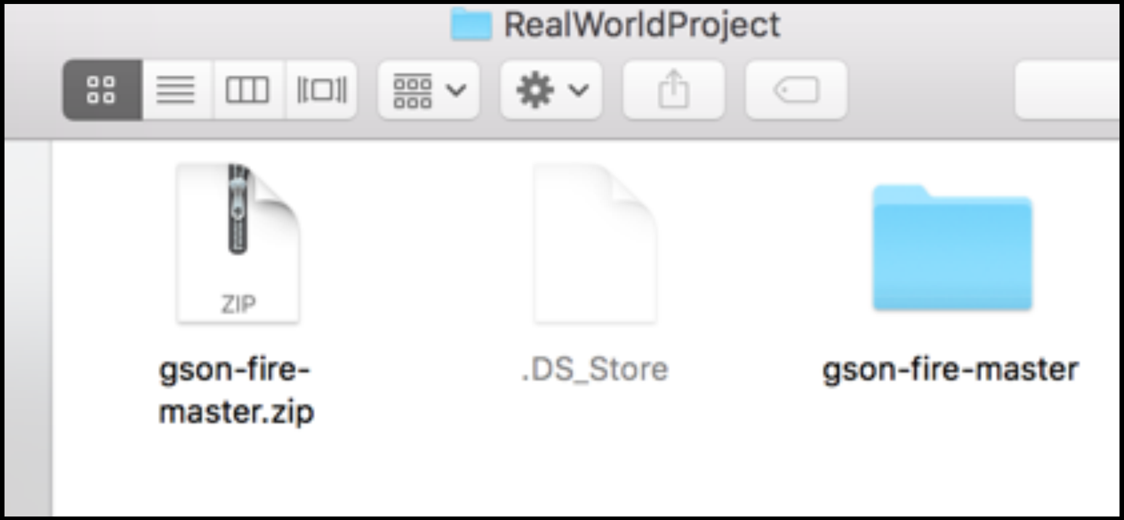
```
https://github.com/julman99/gson-fire.git
```

Open in Desktop Download ZIP

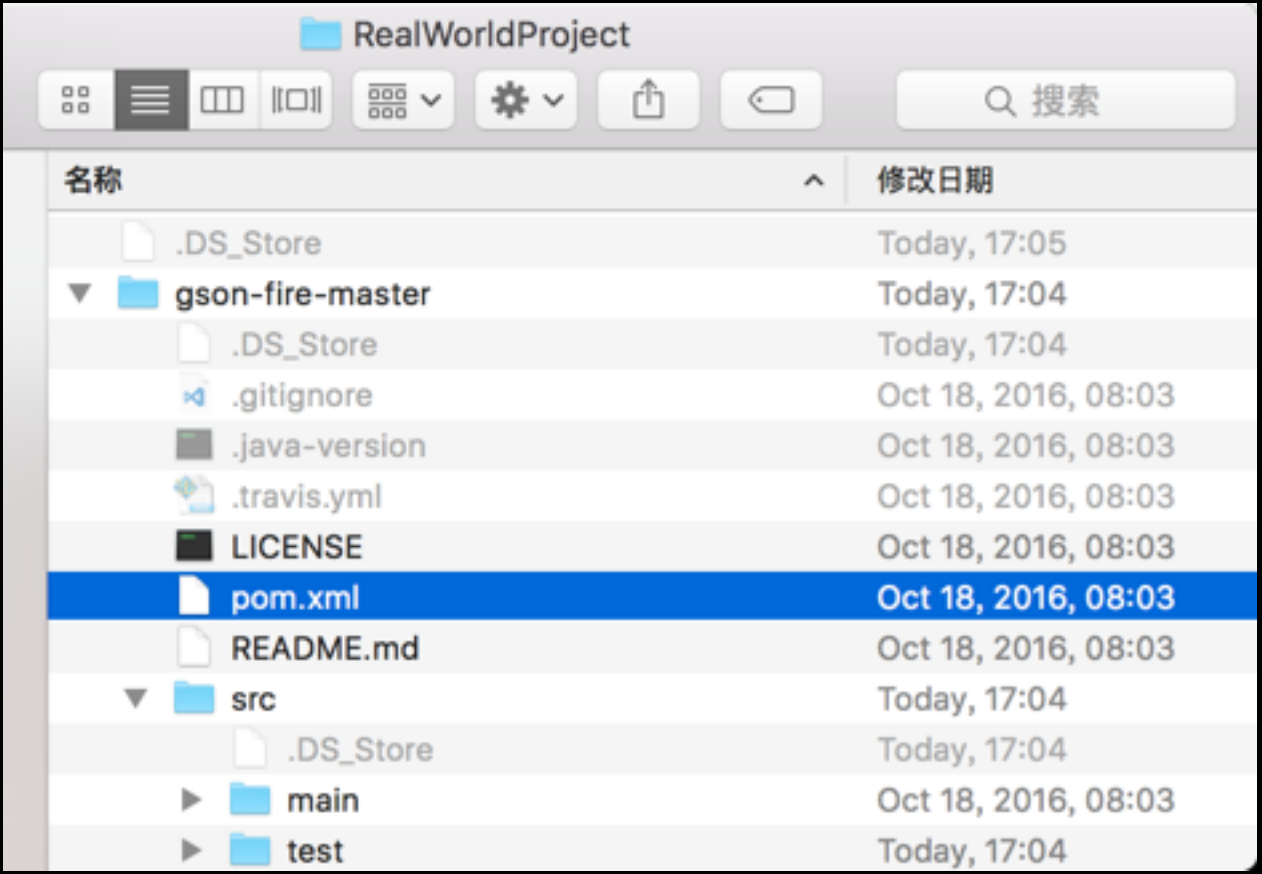
README.md

Team-8-Prioritization-T....zip 显示全部

Put Downloaded gson-fire-master.zip file inside a new directory or any directory (Directory named simple and with no space between words is recommended, such as RealWorldProject), and extract the zip file.



Open the pom.xml file in the root directory of the project

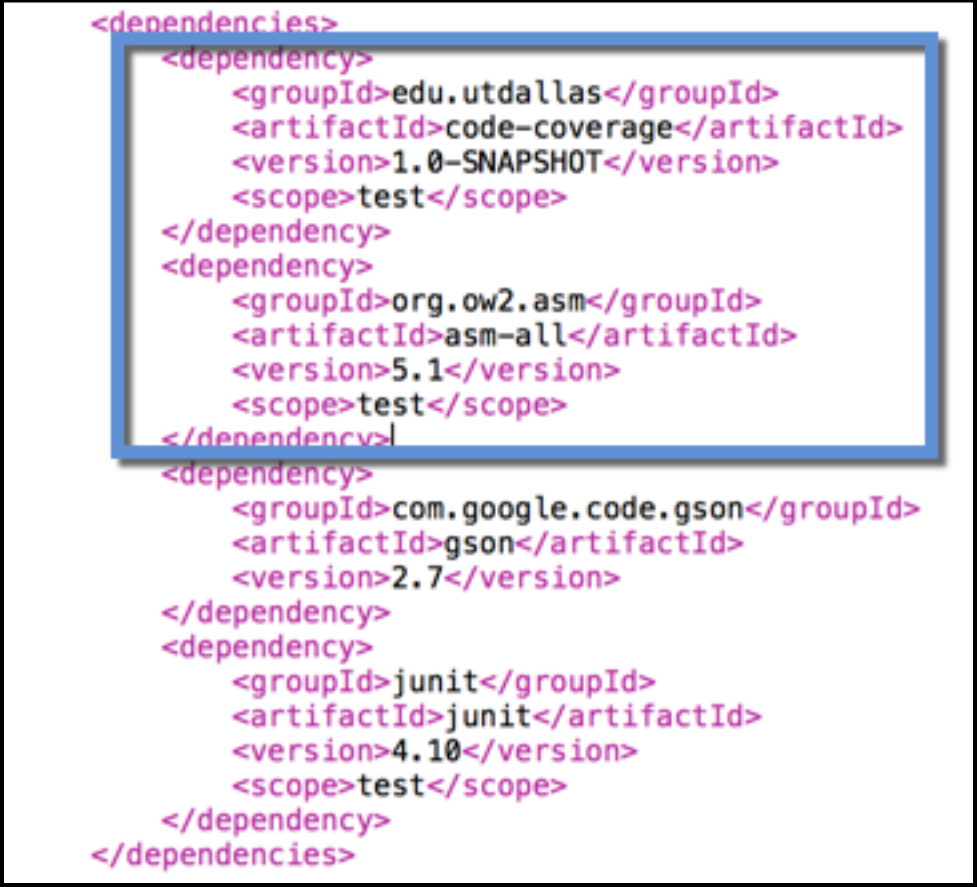


名称	修改日期
.DS_Store	Today, 17:05
gson-fire-master	Today, 17:04
.DS_Store	Today, 17:04
.gitignore	Oct 18, 2016, 08:03
.java-version	Oct 18, 2016, 08:03
.travis.yml	Oct 18, 2016, 08:03
LICENSE	Oct 18, 2016, 08:03
pom.xml	Oct 18, 2016, 08:03
README.md	Oct 18, 2016, 08:03
src	Today, 17:04
.DS_Store	Today, 17:04
main	Oct 18, 2016, 08:03
test	Today, 17:04

First you need to add dependencies
(Copy and paste from below)

```
<dependency>
        <groupId>edu.utdallas</groupId>
        <artifactId>code-coverage</artifactId>
        <version>1.0-SNAPSHOT</version>
        <scope>test</scope>
 </dependency>
 <dependency>
        <groupId>org.ow2.asm</groupId>
        <artifactId>asm-all</artifactId>
        <version>5.1</version>
        <scope>test</scope>
 </dependency>
```

Such as in the example



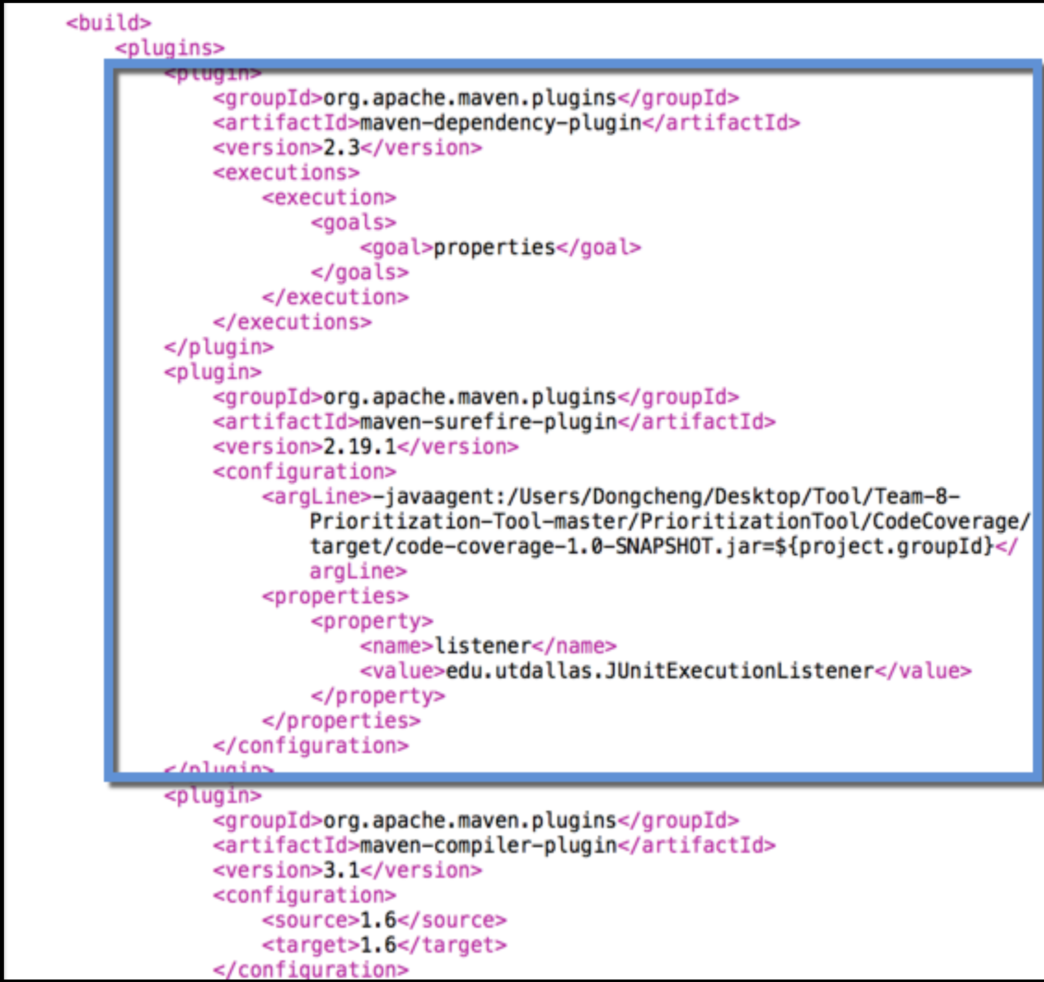
```
<dependencies>
    <dependency>
        <groupId>edu.utdallas</groupId>
        <artifactId>code-coverage</artifactId>
        <version>1.0-SNAPSHOT</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.ow2.asm</groupId>
        <artifactId>asm-all</artifactId>
        <version>5.1</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.7</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.10</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Next you need to add the plugins (Copy and paste from below)

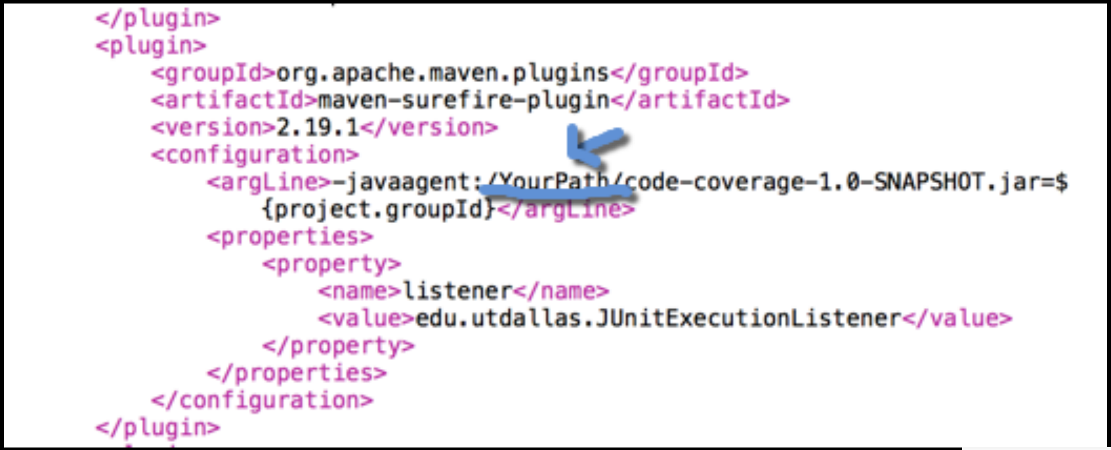
```
<build>
   <plugins>
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-dependency-plugin</artifactId>
         <version>2.3</version>
         <executions>
            <execution>
               <goals>
                  <goal>properties</goal>
               </goals>
            </execution>
         </executions>
                    </plugin>

      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-surefire-plugin</artifactId>
         <version>2.19.1</version>
         <configuration>
         <argLine>-javaagent:/YourPath/code-coverage-1.0-SNAPSHOT.jar=$
{project.groupId}</argLine>
         <properties>
            <property>
               <name>listener</name>
               <value>edu.utdallas.JUnitExecutionListener</value>
            </property>
         </properties>
         </configuration>
      </plugin>
   </plugins>
</build>
```

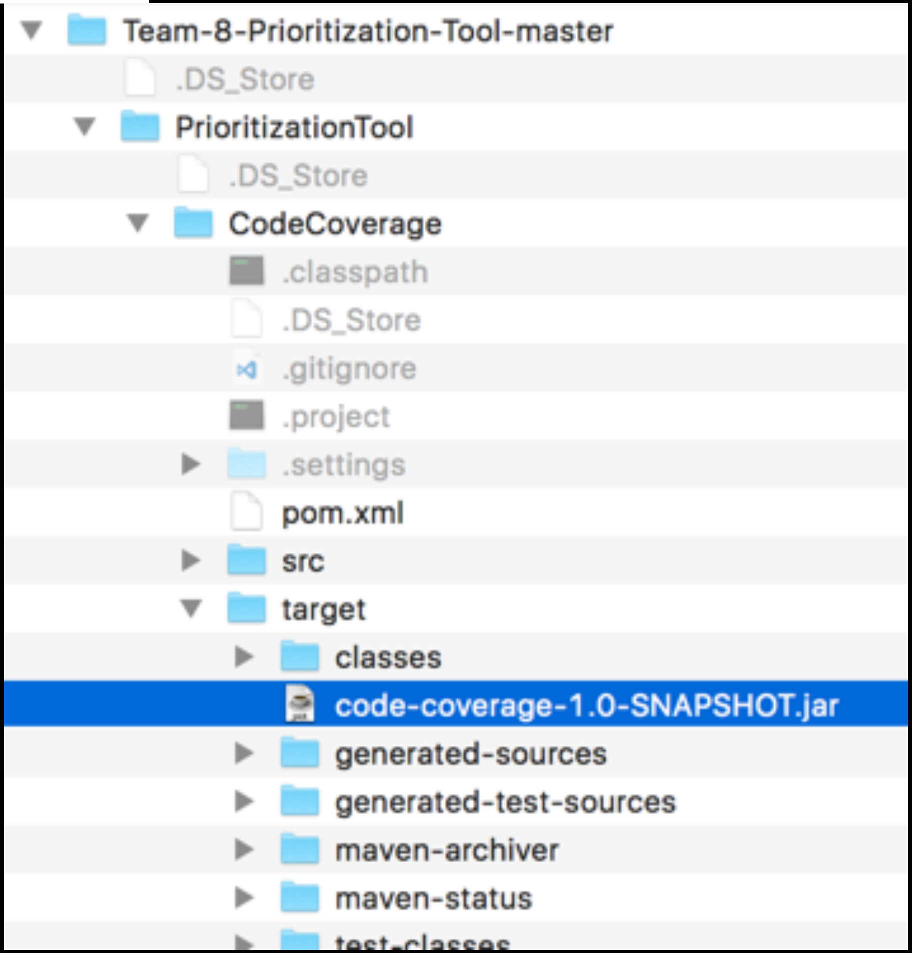
Such as in the example



```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
            <version>2.3</version>
            <executions>
                <execution>
                    <goals>
                        <goal>properties</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.19.1</version>
            <configuration>
                <argLine>-javaagent:/Users/Dongcheng/Desktop/Tool/Team-8-
                    Prioritization-Tool-master/PrioritizationTool/CodeCoverage/
                    target/code-coverage-1.0-SNAPSHOT.jar=${project.groupId}</
                    argLine>
                <properties>
                    <property>
                        <name>listener</name>
                        <value>edu.utdallas.JUnitExecutionListener</value>
                    </property>
                </properties>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.1</version>
            <configuration>
                <source>1.6</source>
                <target>1.6</target>
            </configuration>
```



```
    </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.19.1</version>
        <configuration>
            <argLine>-javaagent:/YourPath/code-coverage-1.0-SNAPSHOT.jar=$
                {project.groupId}</argLine>
            <properties>
                <property>
                    <name>listener</name>
                    <value>edu.utdallas.JUnitExecutionListener</value>
                </property>
            </properties>
        </configuration>
    </plugin>
```



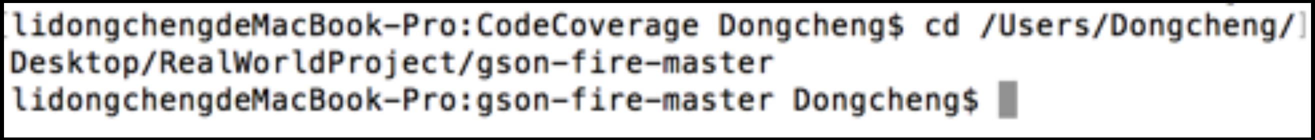
```
▼ Team-8-Prioritization-Tool-master
      .DS_Store
  ▼ PrioritizationTool
        .DS_Store
    ▼ CodeCoverage
          .classpath
          .DS_Store
          .gitignore
          .project
        ▶ .settings
          pom.xml
        ▶ src
        ▼ target
            ▶ classes
              code-coverage-1.0-SNAPSHOT.jar
            ▶ generated-sources
            ▶ generated-test-sources
            ▶ maven-archiver
            ▶ maven-status
            ▶ test-classes
```

The YourPath here, is the path of the code-coverage-1.0-SNAPSHOT.jar file, which we generated by mvn install the PrioritizationTool, this jar file is created under the target directory of the CodeCoverage

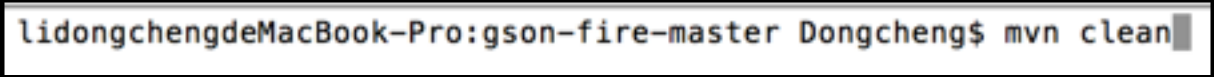
For example, in the example project the YourPath is "/Users/Dongcheng/Desktop/Tool/Team-8-Prioritization-Tool-master/PrioritizationTool/CodeCoverage/target/".

First Run

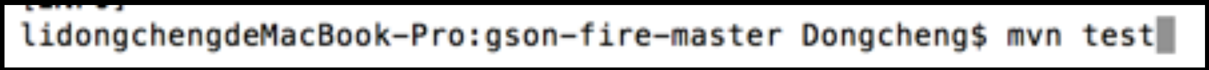
In the Commend Line, Direct to the directory of the pom.xml file inside of the test program, then mvn clean, after that, mvn test



```
lidongchengdeMacBook-Pro:CodeCoverage Dongcheng$ cd /Users/Dongcheng/
Desktop/RealWorldProject/gson-fire-master
lidongchengdeMacBook-Pro:gson-fire-master Dongcheng$
```



```
lidongchengdeMacBook-Pro:gson-fire-master Dongcheng$ mvn clean
```



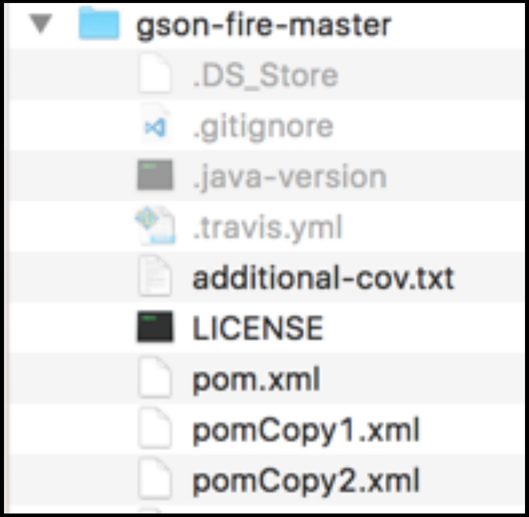
```
lidongchengdeMacBook-Pro:gson-fire-master Dongcheng$ mvn test
```

In the console it will output the default class execution order and calculated total and additional prioritizations for the each test method

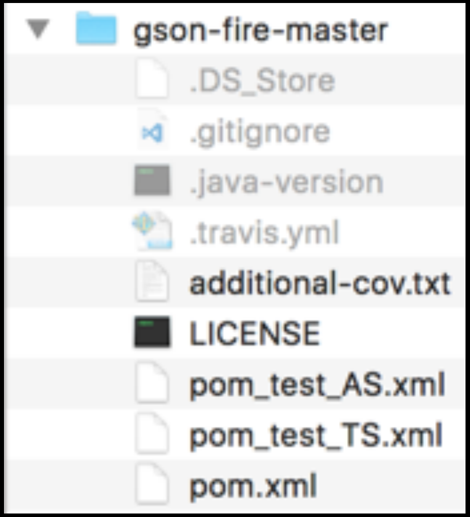
Then it will show build success

Second Run

First make two copies of the pom.xml file, one for Total Strategy and one for Additional strategy, rename them according to the Prioritization strategy they're using
For example, we named pom_test_AS.xml for test program using Additional strategy, and pom_test_TS.xml for test program using Total strategy.



gson-fire-master
.DS_Store
.gitignore
.java-version
.travis.yml
additional-cov.txt
LICENSE
pom.xml
pomCopy1.xml
pomCopy2.xml



gson-fire-master
.DS_Store
.gitignore
.java-version
.travis.yml
additional-cov.txt
LICENSE
pom_test_AS.xml
pom_test_TS.xml
pom.xml

Then we add "Includes" after the beginning of the <configuration> tag or right before the <argLine> tag, inside of the "maven-surefire-plugin", add to the pom file according to the strategy its using

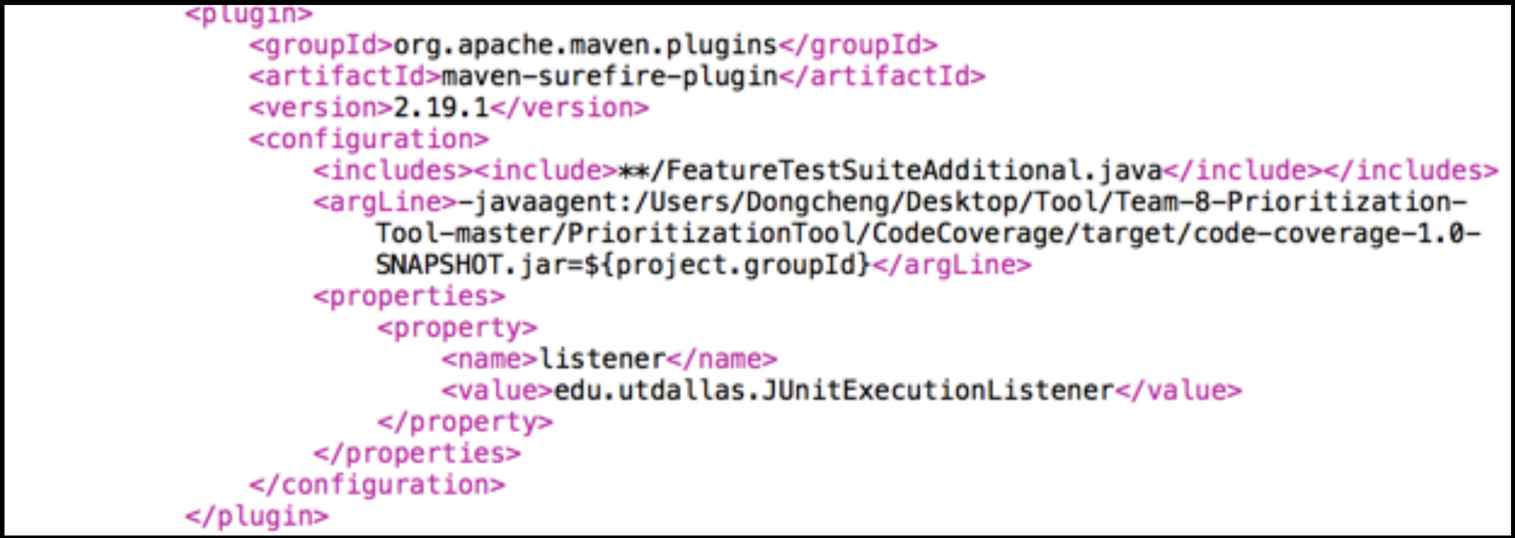
If using total strategy, add:

<includes><include>**/FeatureTestSuite.java</include></includes>

If using additional strategy, add:

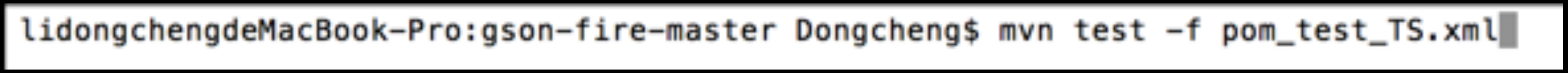
<includes><include>**/FeatureTestSuiteAdditional.java</include></includes>

Such as in the test program, we add
<includes><include>**/FeatureTestSuite.java</include></includes>
In the pom_test_TS.xml, and we add
<includes><include>**/FeatureTestSuite.java</include></includes>
In the pom_test_TS.xml

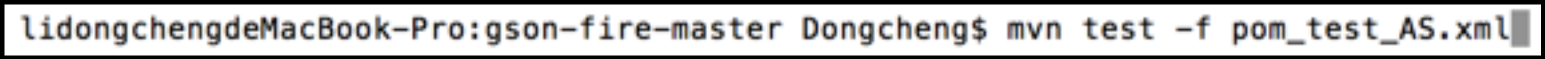


```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.19.1</version>
    <configuration>
        <includes><include>**/FeatureTestSuiteAdditional.java</include></includes>
        <argLine>-javaagent:/Users/Dongcheng/Desktop/Tool/Team-8-Prioritization-
            Tool-master/PrioritizationTool/CodeCoverage/target/code-coverage-1.0-
            SNAPSHOT.jar=${project.groupId}</argLine>
        <properties>
            <property>
                <name>listener</name>
                <value>edu.utdallas.JUnitExecutionListener</value>
            </property>
        </properties>
    </configuration>
</plugin>
```

Finally, if we want to test program using Total strategy, under the directory of the test program, we just type mvn test -f pom_test_TS.xml or if we want to test program using Additional strategy we just type mvn test -f pom_test_AS.xml



```
lidongchengdeMacBook-Pro:gson-fire-master Dongcheng$ mvn test -f pom_test_TS.xml
```

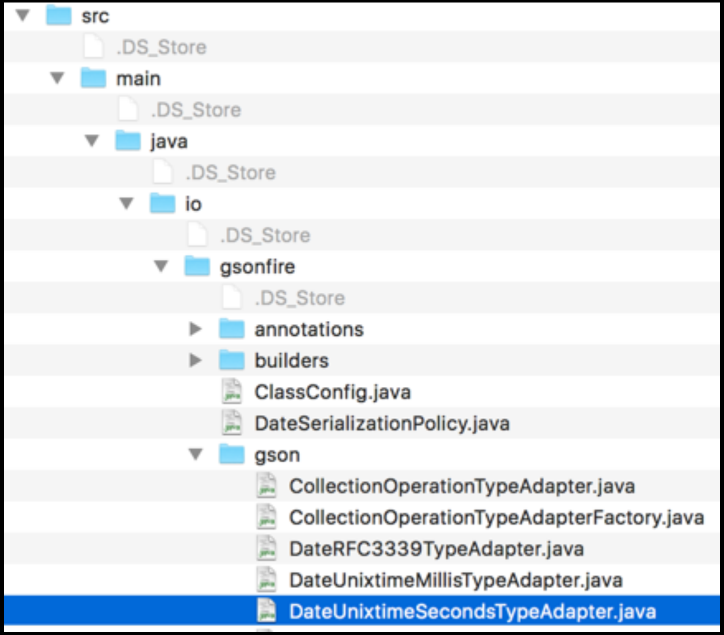


```
lidongchengdeMacBook-Pro:gson-fire-master Dongcheng$ mvn test -f pom_test_AS.xml
```

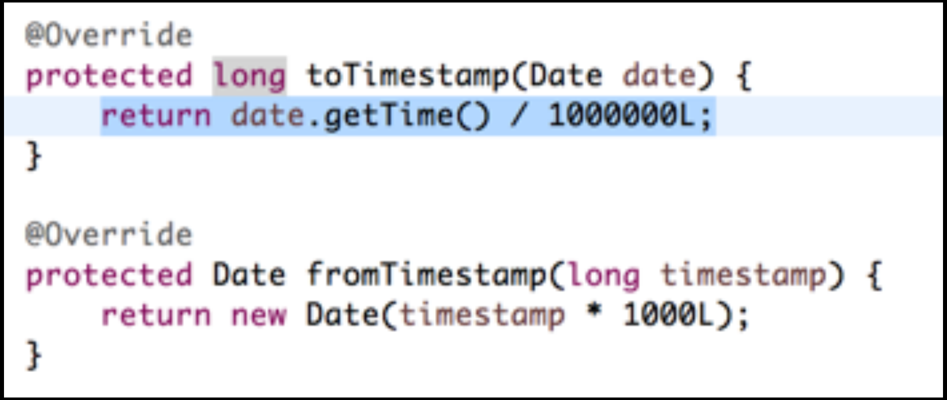
In the console it will output the manipulated class execution order using the choice of user

To compare the time to detect the first bug of the original execution and the prioritized execution. First, we manually create bug for the test program.

- Such as in the test program, gson-fire, under src/main/java/io/gsonfire/gson in the *DateUnixtimeSecondsTypeAdapter* class, we increase the value of the denominator of the return value from 1000L to 1000000L. It causes one failure during the program execution.



```
▼ src
    .DS_Store
  ▼ main
      .DS_Store
    ▼ java
        .DS_Store
      ▼ io
          .DS_Store
        ▼ gsonfire
            .DS_Store
          ► annotations
          ► builders
            ClassConfig.java
            DateSerializationPolicy.java
          ▼ gson
              CollectionOperationTypeAdapter.java
              CollectionOperationTypeAdapterFactory.java
              DateRFC3339TypeAdapter.java
              DateUnixtimeMillisTypeAdapter.java
              DateUnixtimeSecondsTypeAdapter.java
```



```
@Override
protected long toTimestamp(Date date) {
    return date.getTime() / 1000000L;
}

@Override
protected Date fromTimestamp(long timestamp) {
    return new Date(timestamp * 1000L);
}
```

Then, just run the program with
mvn test , for the original execution
mvn test -f pom_test_TS.xml , or
mvn test -f pom_test_TS.xml , for the prioritized execution.