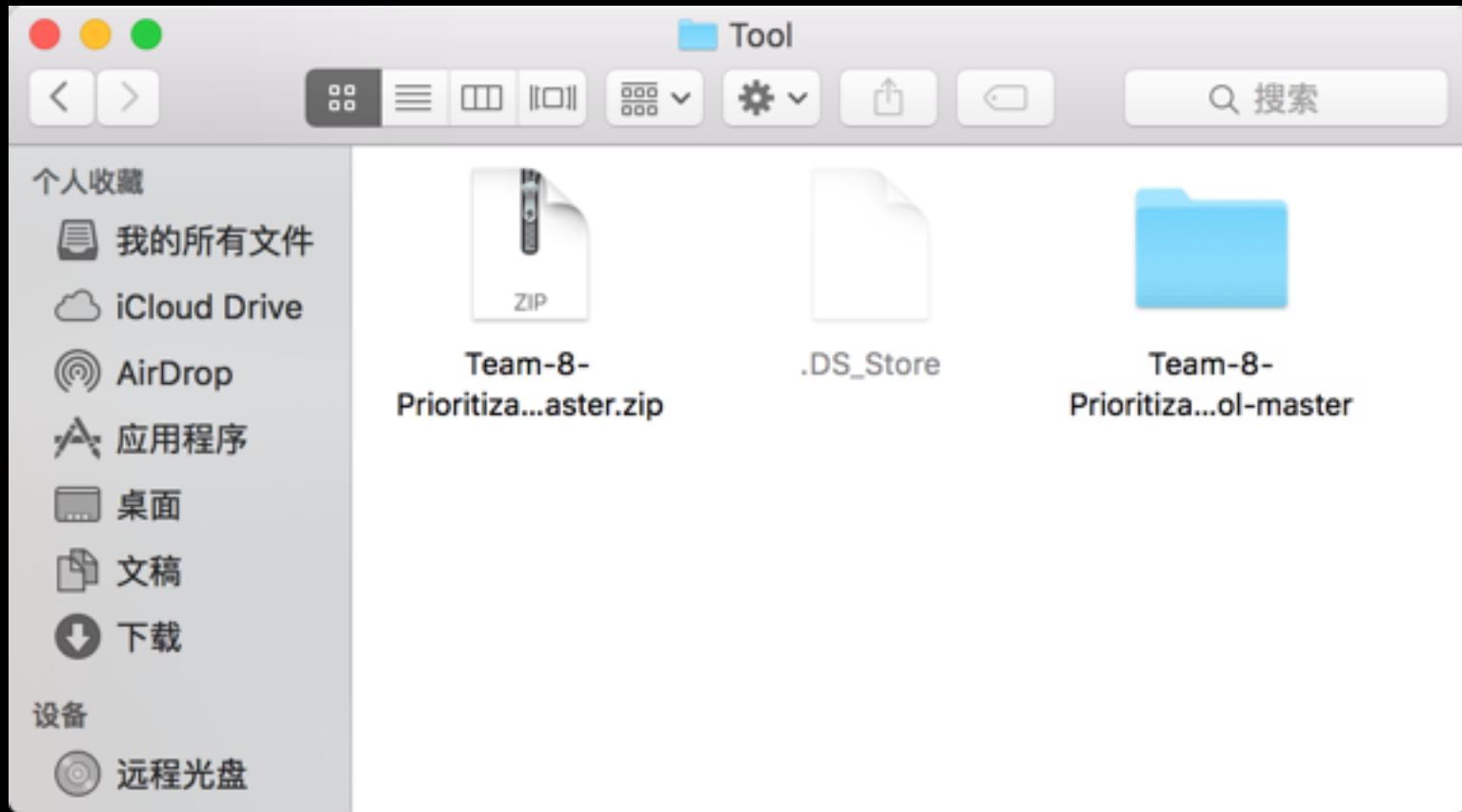


# Download Prioritization Tool from the GitHub repository, <https://github.com/dongchengli/Team-8-Prioritization-Tool>

The screenshot shows the GitHub repository page for 'dongchengli / Team-8-Prioritization-Tool'. The repository is private and has 5 commits, 1 branch, 0 releases, and 1 contributor. The 'Code' tab is selected, showing the file structure with 'PrioritizationTool/CodeCoverage' and 'README.md'. A 'Clone or download' dropdown menu is open, showing the 'Clone with HTTPS' option with the URL 'https://github.com/dongchengli/Team-8-Prio'. The 'README.md' file is open, displaying the text: '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 JavaAgent class, then override "visitLine" method in "MethodTransformVisitor" class.', and '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.

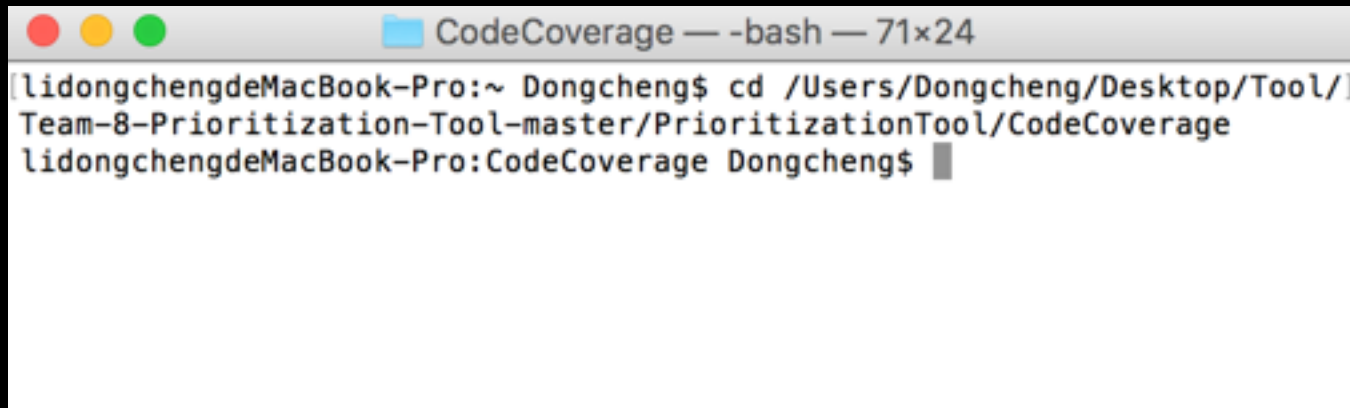


From the command line, first check the maven, if no maven installed, it need to be installed

A screenshot of a macOS terminal window. The title bar shows three colored window control buttons (red, yellow, green) on the left, a home icon and the text 'Dongcheng — -bash — 85x27' in the center. The terminal content shows the command 'mvn -v' being executed, which outputs the following information: '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', and 'OS name: "mac os x", version: "10.11.5", arch: "x86\_64", family: "mac"'. The prompt 'lidongchengdeMacBook-Pro:~ Dongcheng\$' is visible at the end of the output.

```
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 Command Line Direct to the directory of the CodeCoverage folder inside of the Team-8-Prioritization-Tool-master folder

A screenshot of a macOS terminal window. The title bar shows three colored window control buttons (red, yellow, green) on the left, followed by a blue folder icon and the text "CodeCoverage — -bash — 71x24". The terminal text shows the user navigating to a specific directory. The prompt is "lidongchengdeMacBook-Pro:~ Dongcheng\$". The command entered is "cd /Users/Dongcheng/Desktop/Tool/Team-8-Prioritization-Tool-master/PrioritizationTool/CodeCoverage". The next line shows the prompt has changed to "lidongchengdeMacBook-Pro:CodeCoverage Dongcheng\$".

```
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

A screenshot of a terminal window showing the output of a Maven build. The output consists of several lines of text, each preceded by "[INFO]". The first line is a separator line of dashes. The second line says "BUILD SUCCESS". The third line is another separator line of dashes. The fourth line says "Total time: 2.195 s". The fifth line says "Finished at: 2017-04-29T16:46:42-05:00". The sixth line says "Final Memory: 17M/213M". The seventh line is a final separator line of dashes. The prompt at the bottom is "lidongchengdeMacBook-Pro:CodeCoverage Dongcheng\$".

```
[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.

The screenshot shows the GitHub repository page for `julman99/gson-fire`. The repository is described as "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!" with a link to <http://gsonfire.io>.

Repository statistics:

- 187 commits
- 17 branches
- 10 releases
- 5 contributors

Branch: master | New pull request

Buttons: Create new file, Upload files, Find file, Clone or download

Clone with HTTPS (selected) | Use SSH

Use Git or checkout with SVN using the web URL.

<https://github.com/julman99/gson-fire.git>

Open in Desktop | Download ZIP

File list:

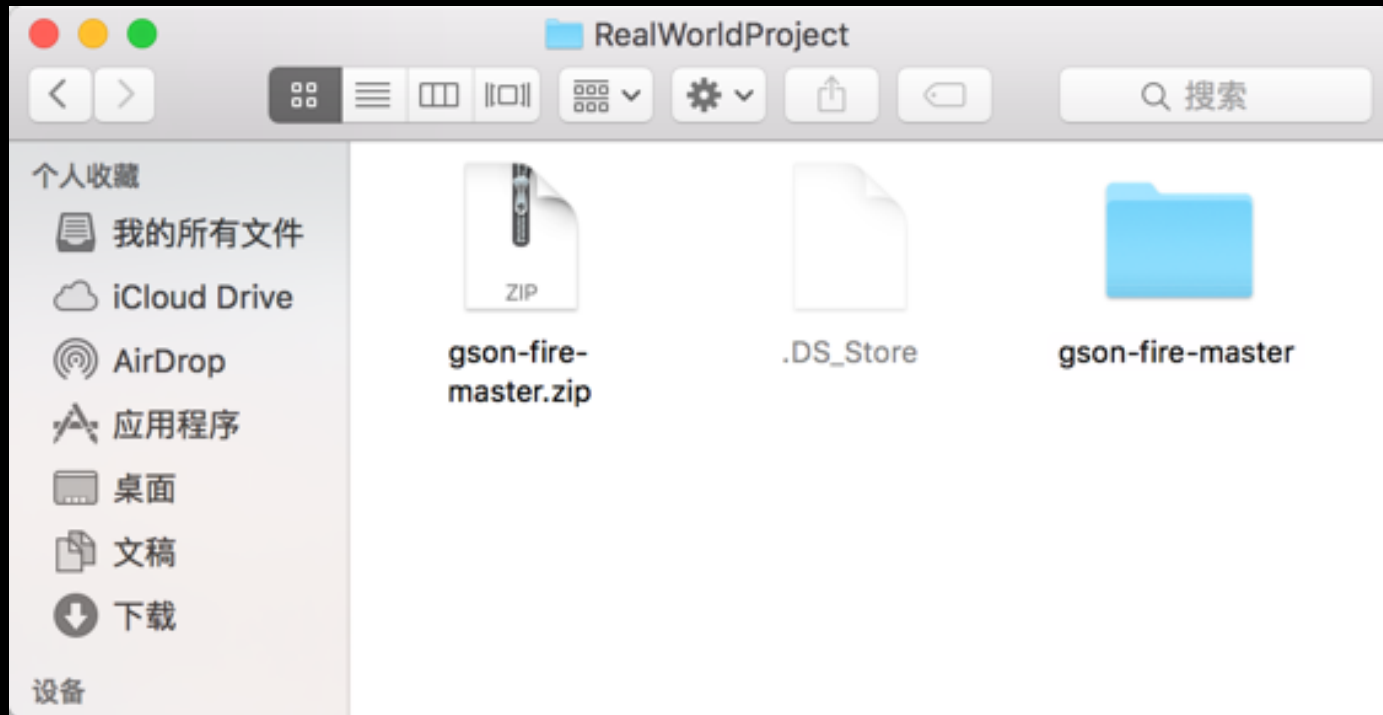
File	Description	Last Commit
src	Ensures we clean alreadyResolvedTypeTokensRegistry even on Exce	
.gitignore	Initial commit	
.java-version	Version up	
.travis.yml	Adds travis file	
LICENSE	Update copyright year	2 years ago
README.md	Version up: 1.8.0	6 months ago
pom.xml	Version up: 1.8.0	6 months ago

README.md

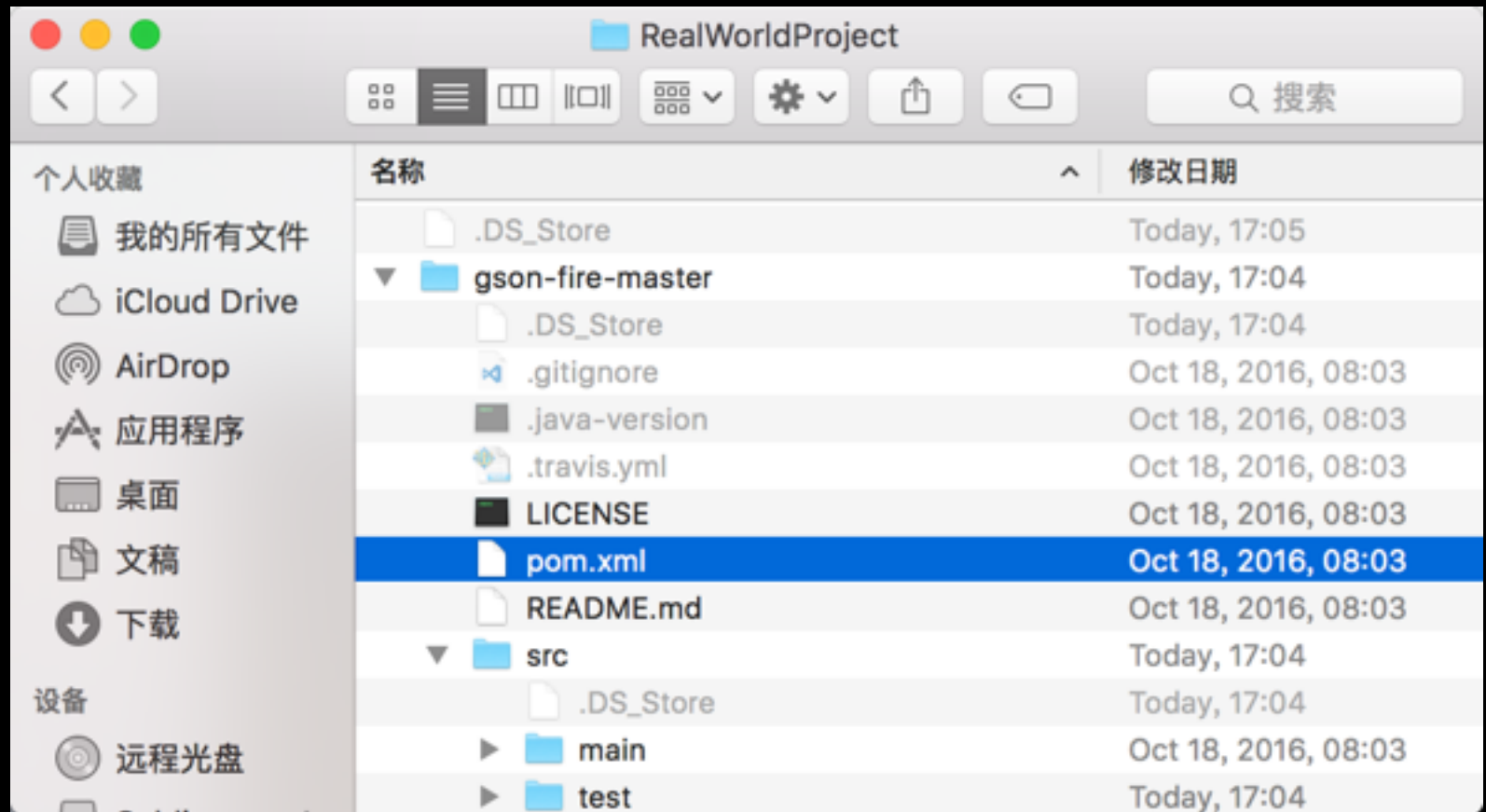
Team-8-Prioritization-T....zip

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



First you need to add dependencies

Such as in the example

```
<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>
```

```
<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

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:/YourPath/code-coverage-1.0-SNAPSHOT.jar=${
{project.groupId}}</argLine>
      <properties>
        <property>
          <name>listener</name>
          <value>edu.utdallas.JUnitExecutionListener</value>
        </property>
      </properties>
    </configuration>
  </plugins>
</build>
```

```
<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>
  </plugins>
```

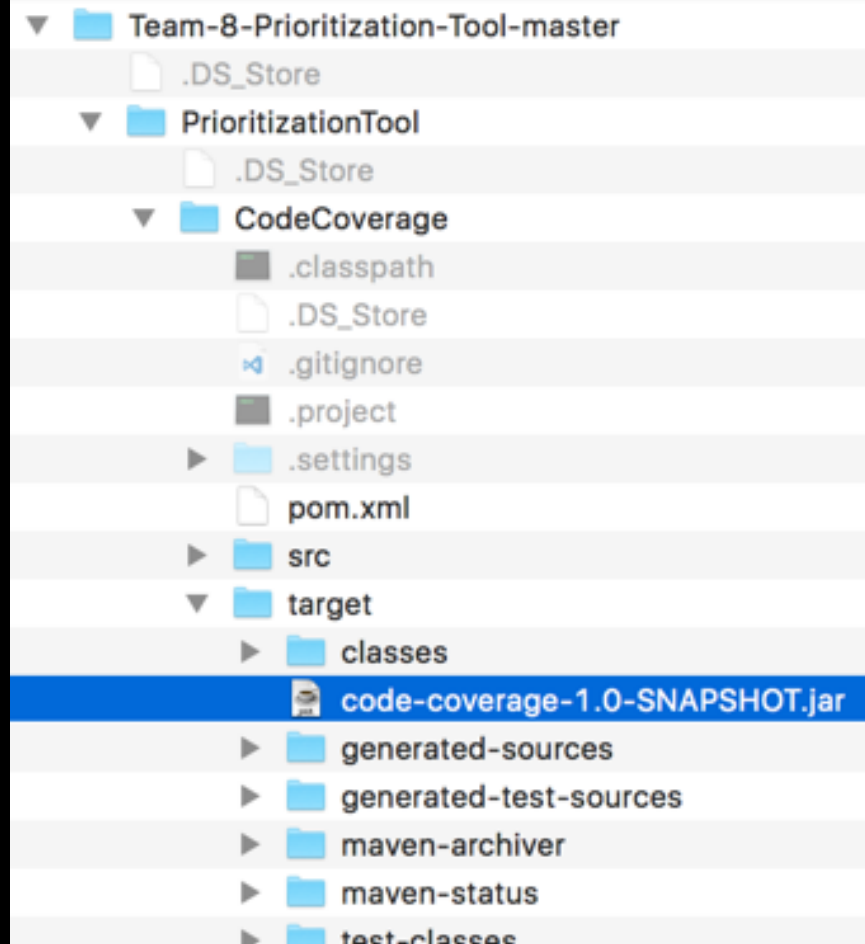
```

</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>

```

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 realworld project the YourPath is “/Users/Dongcheng/Desktop/Tool/Team-8-Prioritization-Tool-master/PrioritizationTool/CodeCoverage/target/”.



In the Command 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
```

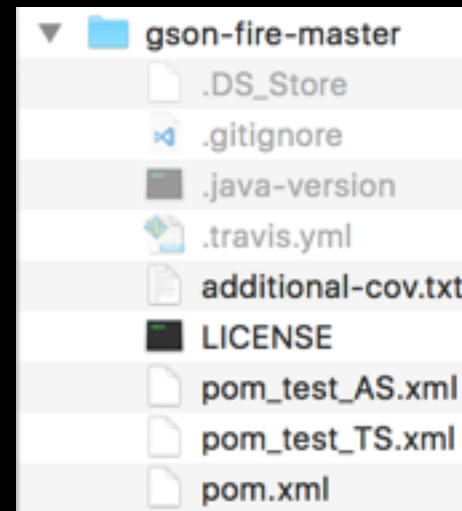
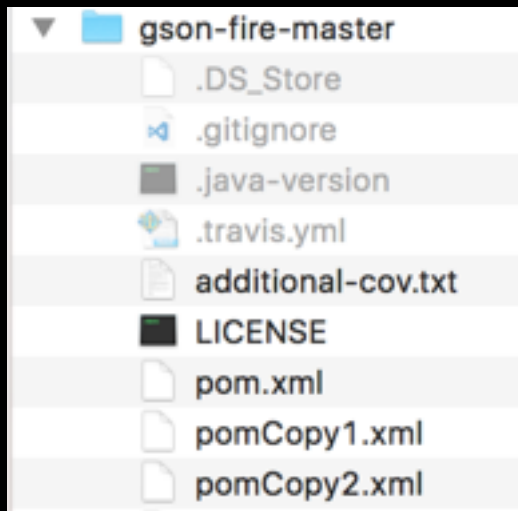
In the console it will output the default class execution order and calculated total and additional prioritizations for the each test method

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, in the we name pom\_test\_AS for test program using Additional strategy, and pom\_test\_TS for test program using Total strategy.



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:

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

If using additional strategy:

```
<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

```
<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