Міністерство освіти і науки України

Запорізький національний технічний університет

кафедра програмних засобів

РЕФЕРАТ

з дисципліни: «Якість програмного забезпечення»

на тему: «СТАТИЧНІ МЕТОДИ ТЕСТУВАННЯ НА ПРИКЛАДІ JMETER APACHE»

Виконав:

студент групи КНТ-115 Е. В. Гунько

Перевірила: Г. В. Табунщик

Запоріжжя

2018

**ЗМІСТ**

[**ЗМІСТ** 2](#_Toc532151243)

[**1.** **ОСНОВНЫЕ ПОНЯТИЯ О МЕТРИКАХ** 3](#_Toc532151244)

[**2.** **СТАТИЧЕСКИЕ МАРШРУТЫ ВНУТРИ КЛАССОВ И ИХ ЗАВИСИМОСТИ** 7](#_Toc532151247)

[**3.** **МАТРИЦА ВЫЗОВОВ** 23](#_Toc532151248)

[**4.** **ВЫВОД** 25](#_Toc532151249)

[**5.** **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 26](#_Toc532151250)

1. **ОСНОВНЫЕ ПОНЯТИЯ О МЕТРИКАХ**

Метрика программного обеспечения (англ. software metric) — мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций. [1]

Метрики

|  |  |
| --- | --- |
| Blank lines | Пустые строки |
| Classes | Классы |
| Code Lines | Строки кода |
| Comment Lines | Строки с комментариями |
| Comment to Code Ratio | Соотношение комментариев к коду |
| Declarative Statements | Операторы объявления |
| Executable Statements | Операторы выполнения |
| Files | Файлы |
| Functions | Функции |
| Lines | Строки |

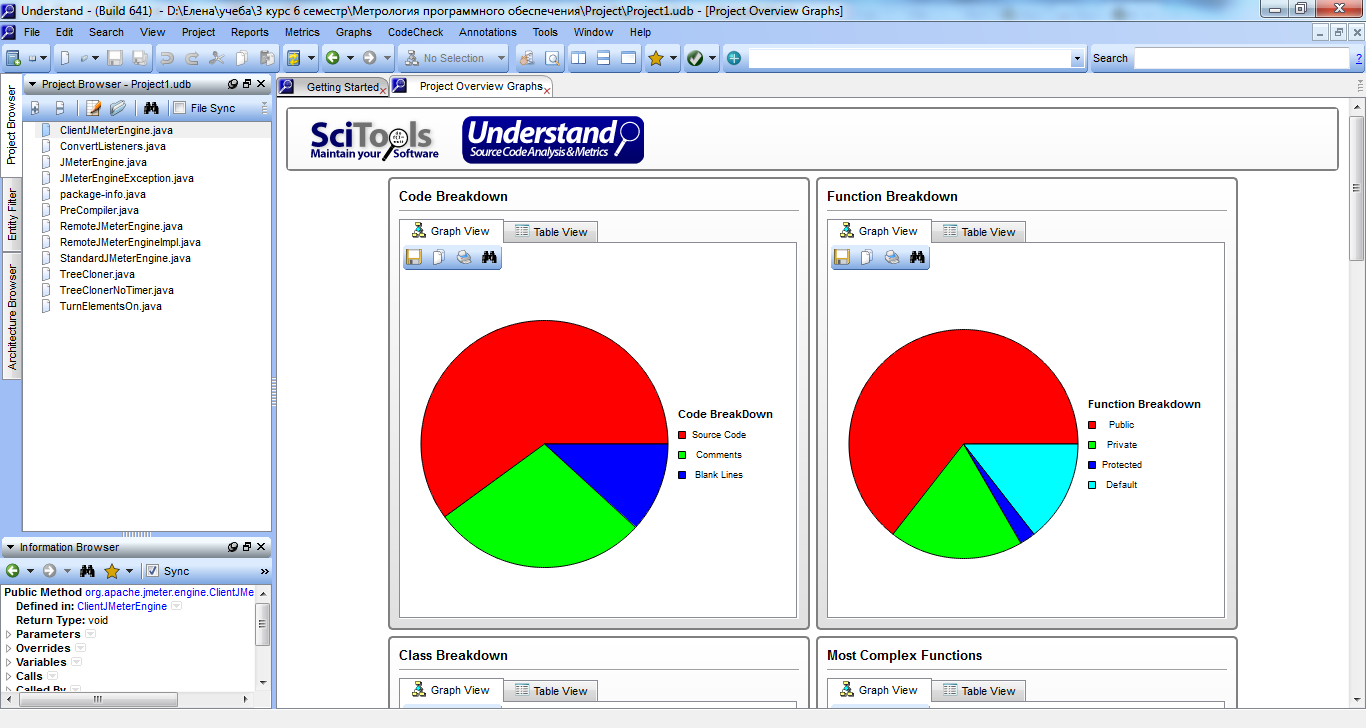
## Количественные метрики

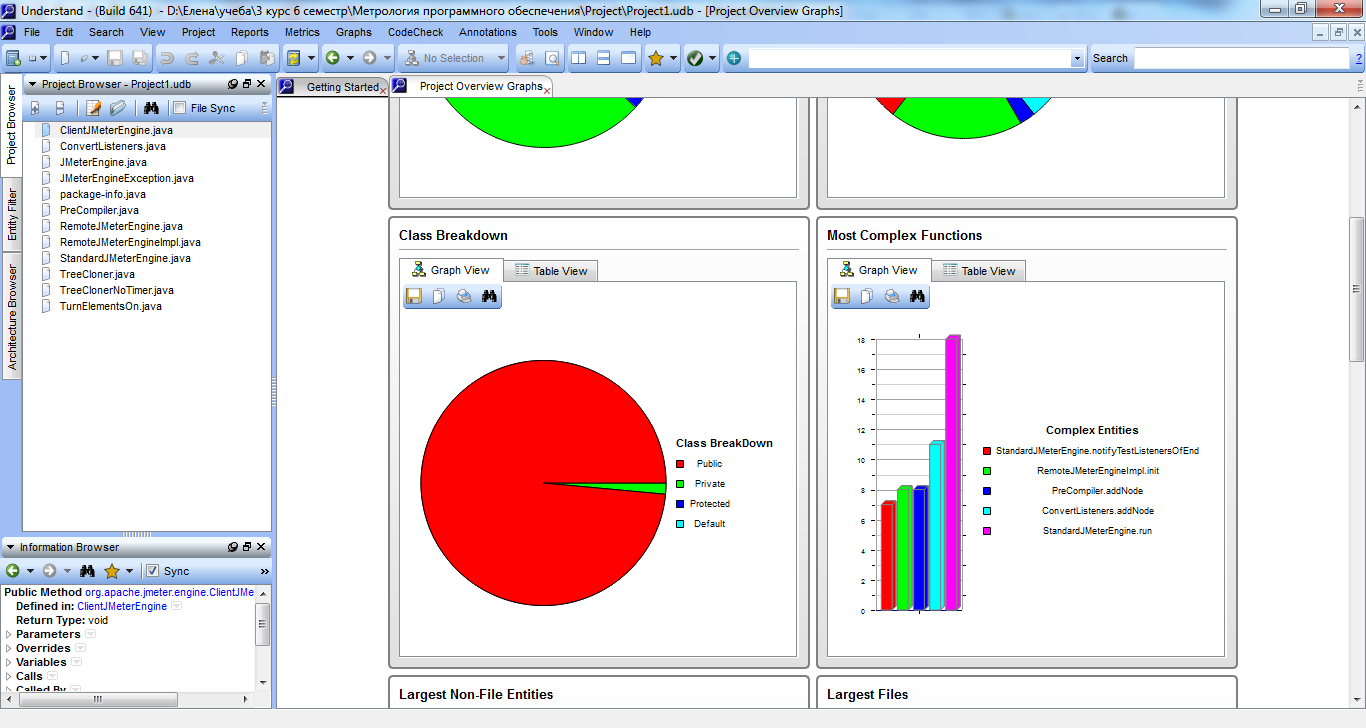
Прежде всего, следует рассмотреть количественные характеристики исходного кода программ (в виду их простоты). Самой элементарной метрикой является количество строк кода (SLOC). Данная метрика была изначально разработана для оценки трудозатрат по проекту. Однако из-за того, что одна и та же функциональность может быть разбита на несколько строк или записана в одну строку, метрика стала практически неприменимой с появлением языков, в которых в одну строку может быть записано больше одной команды. Поэтому различают логические и физические строки кода. Логические строки кода — это количество команд программы. Данный вариант описания так же имеет свои недостатки, так как сильно зависит от используемого языка программирования и стиля программирования [2].

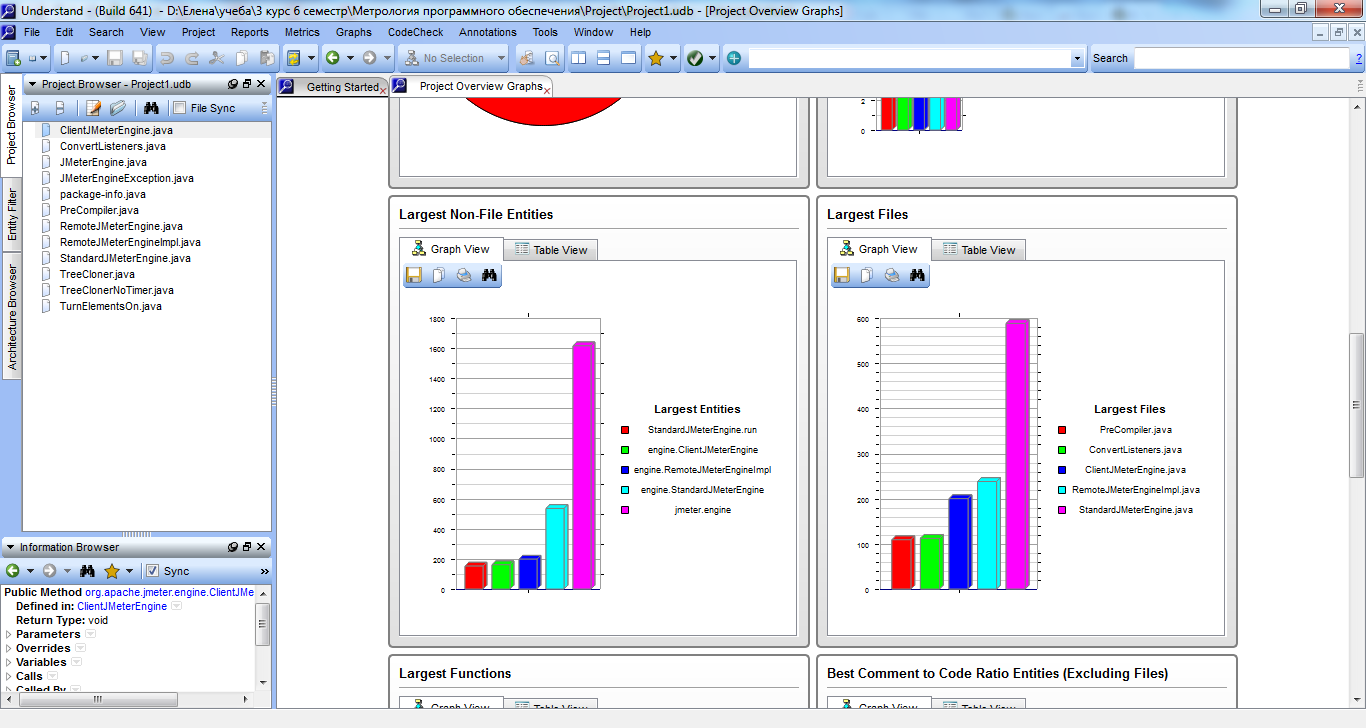
Кроме SLOC к количественным характеристикам относят также:

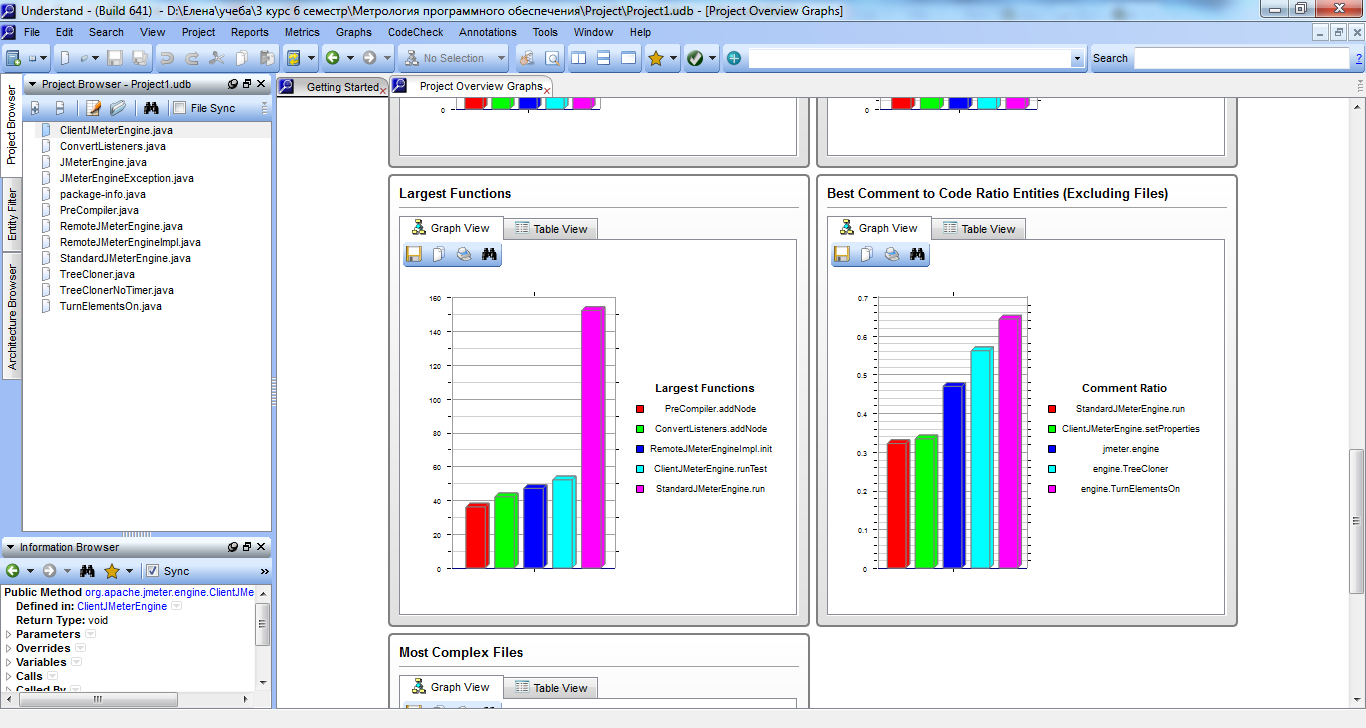
* количество пустых строк,
* количество комментариев,
* процент комментариев (отношение числа строк, содержащих комментарии к общему количеству строк, выраженное в процентах),
* среднее число строк для функций (классов, файлов),
* среднее число строк, содержащих исходный код для функций (классов, файлов),
* среднее число строк для модулей.

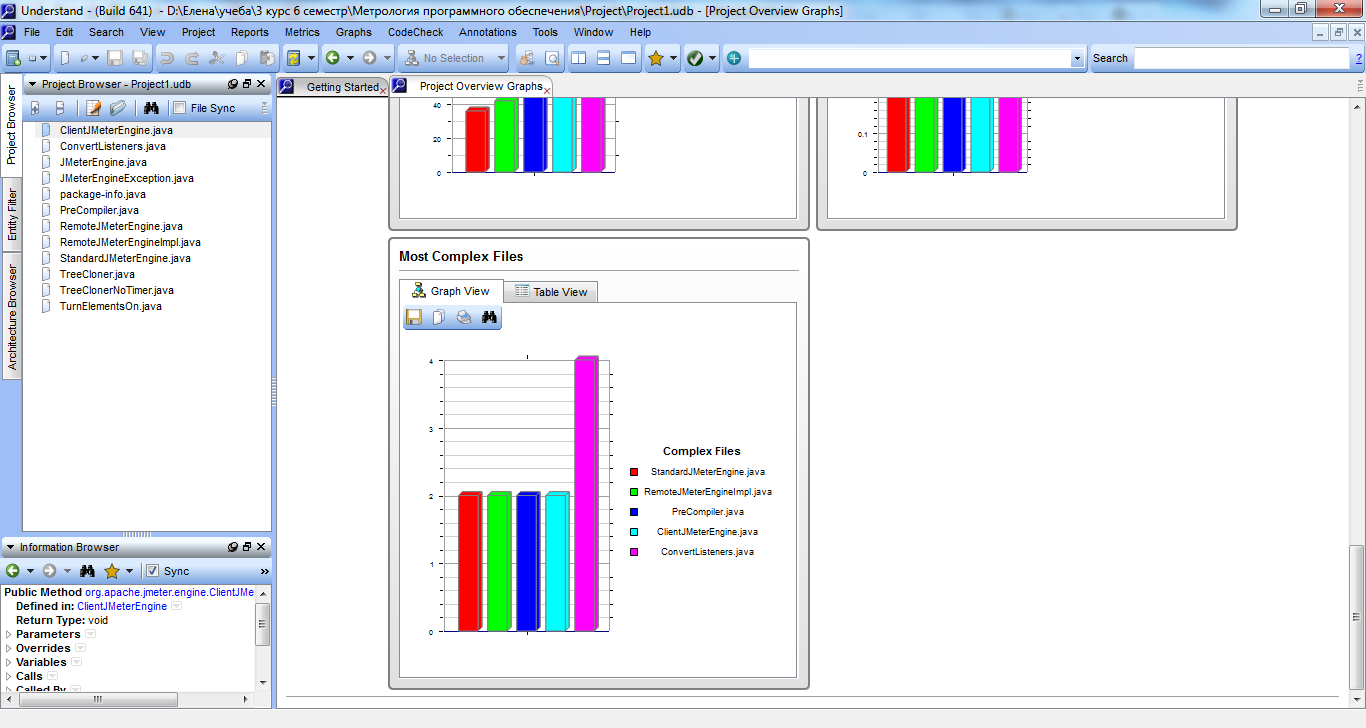
Основные метрики проекта в графическом виде











Метрики предоставляют статистическую информацию о вашем проекте, такие как количество строк кода и сложность различных сущностей[3]. Стандартные метрики, которые могут быть созданы, в зависимости от языка, анализируются включают в себя[4]:

**Основные показатели**

|  |  |
| --- | --- |
| * Класс Count | * Файлы Count |
| * Функция Count | * Соотношение комментарий к Коду |
| * Комментарии линии Count | * Исполняемого Оператора Count |
| * Код Строк Count | * Декларативное Заявление Count |

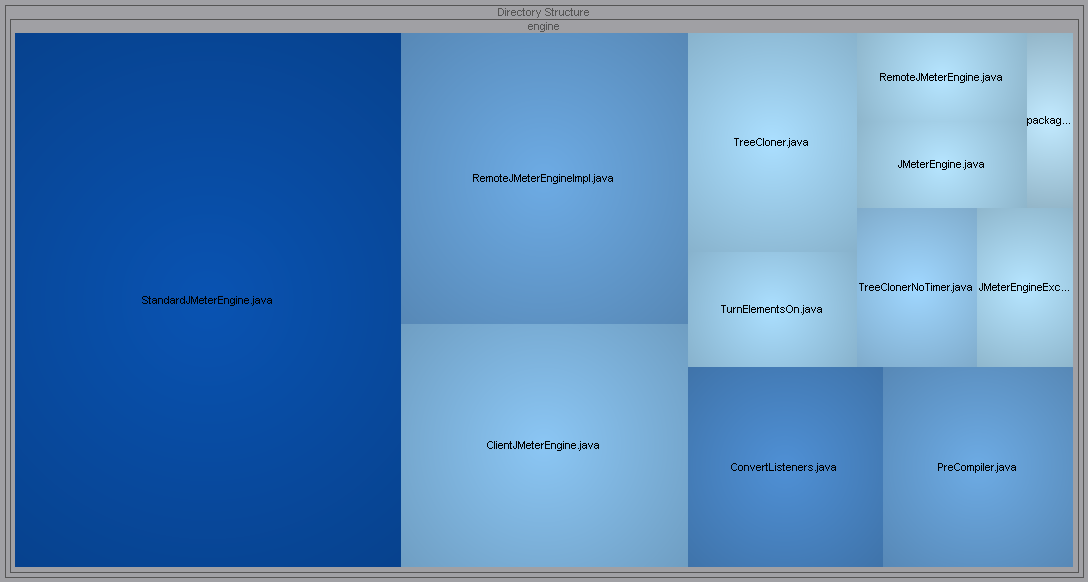
### ****Настраиваемые показатели****

|  |  |
| --- | --- |
| * Цикломатическая Сложность | * Соединение Классов |
| * Путь Count | * Процент Отсутствие Сплоченности |
| * Базовый Класс Count | * Количество Методов Экземпляра |
| * Макс Наследования | * Взвешенный Методы в Классе |
| * Узлы | * Унаследованный Класс Count |

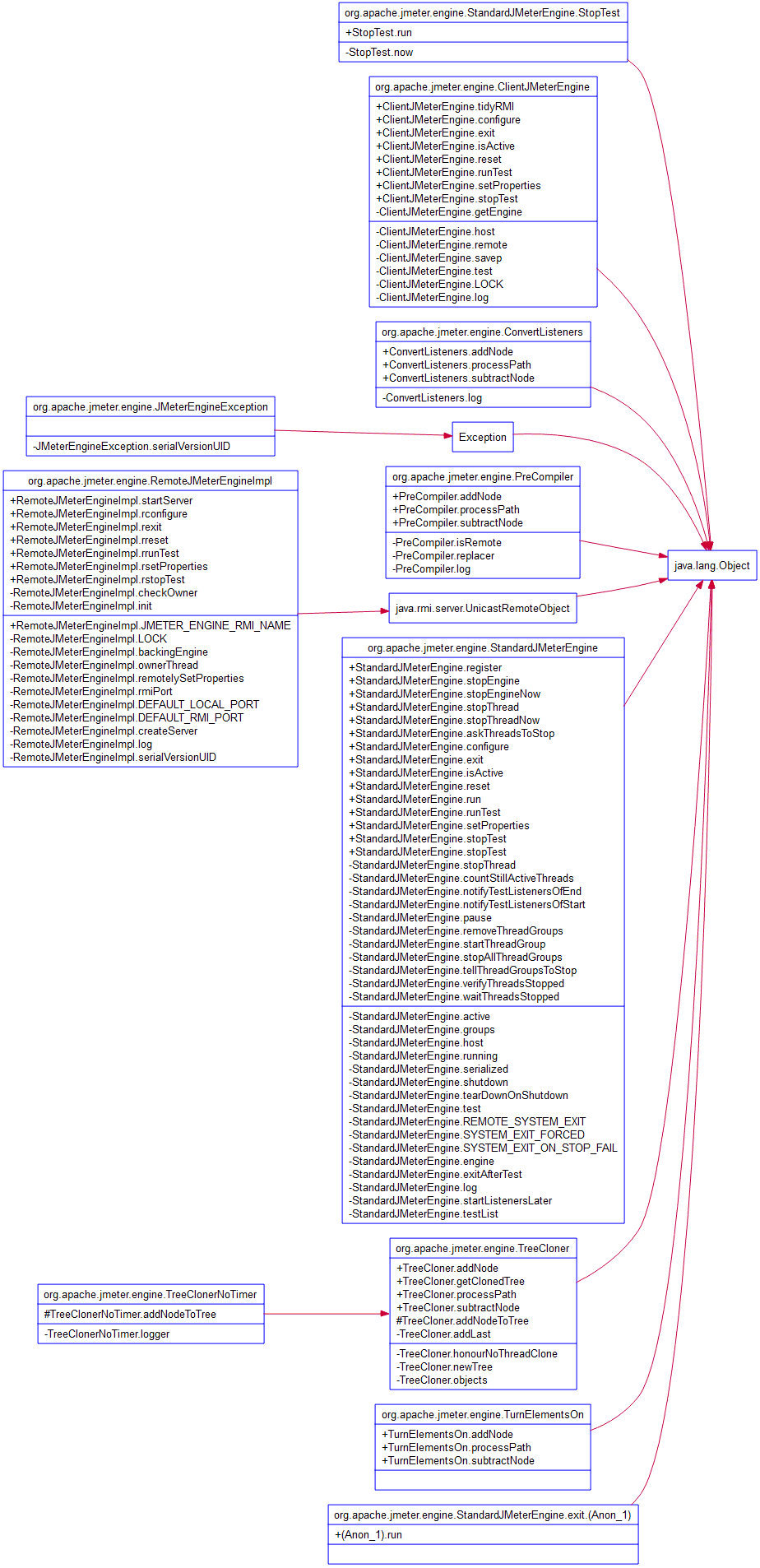
Исследование различных графиков вызовов между классами внутри пакета (между классами).

1. **СТАТИЧЕСКИЕ МАРШРУТЫ ВНУТРИ КЛАССОВ И ИХ ЗАВИСИМОСТИ**

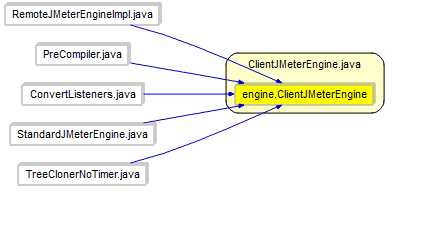
Структура пакетов



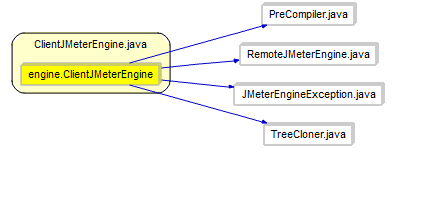
Статические маршруты внутри отдельных классов



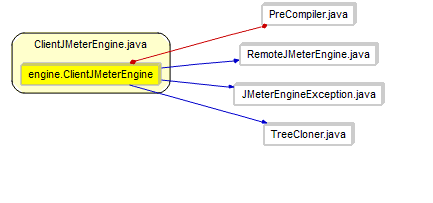
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.ClientMeterEngine (Класс 1)**



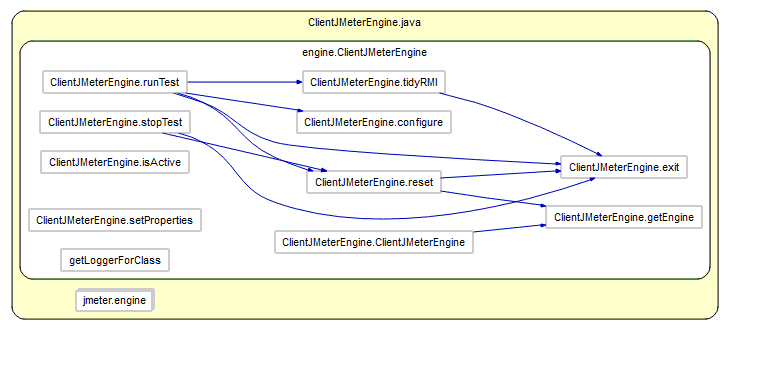
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.ClientMeterEngine



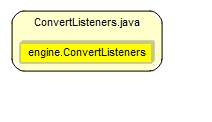
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.ClientMeterEngine и какие классы вызывает сам класс



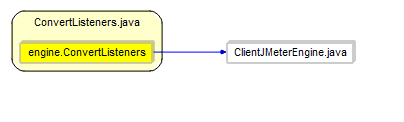
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.ClientMeterEngine



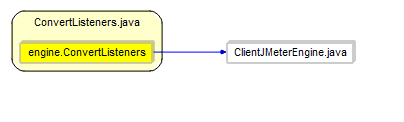
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.ConvertListeners (Класс 2)**



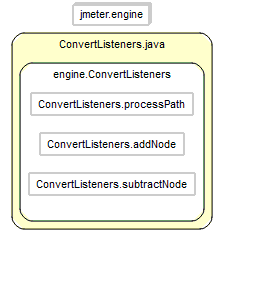
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.ConvertListeners



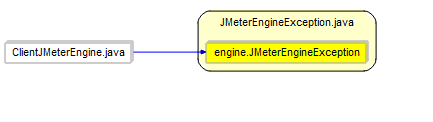
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.ConvertListeners и какие классы вызывает сам класс



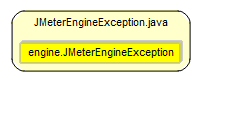
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.ConvertListeners



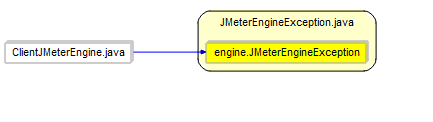
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.JMeterEngineException (Класс 3)**



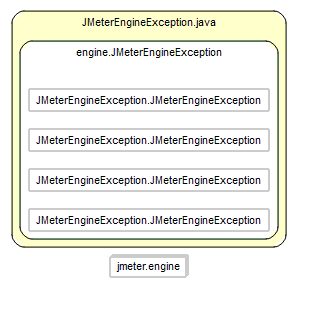
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.JMeterEngineException



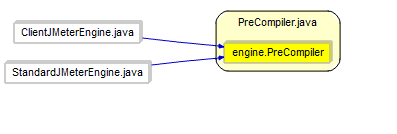
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.JMeterEngineException и какие классы вызывает сам класс



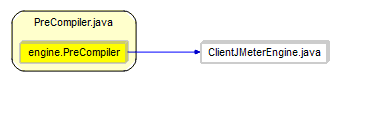
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.JMeterEngineException



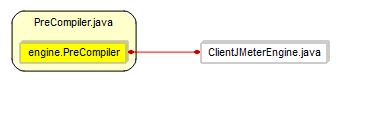
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.PreCompiler (Класс 4)**



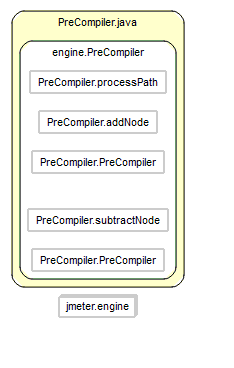
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.PreCompiler



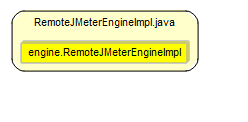
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.PreCompiler и какие классы вызывает сам класс



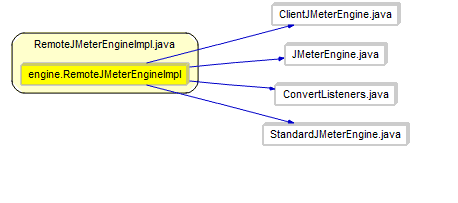
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.PreCompiler



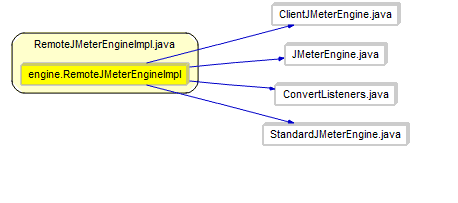
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.RemoteJMeterEngineImpl (Класс 5)**



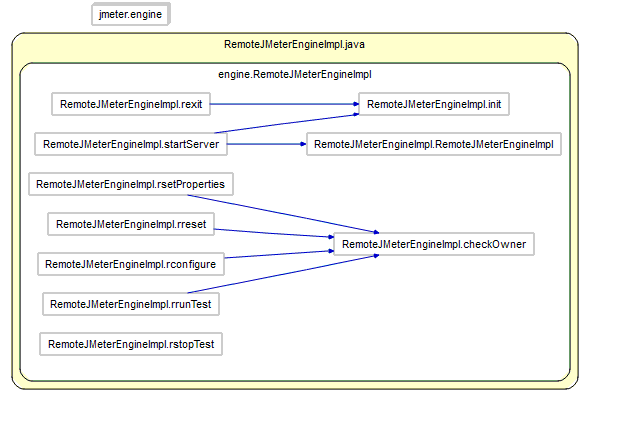
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.RemoteJMeterEngineImpl



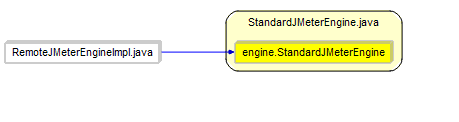
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.RemoteJMeterEngineImpl и какие классы вызывает сам класс



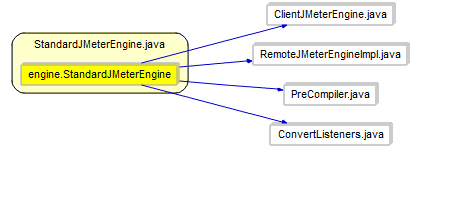
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.RemoteJMeterEngineImpl



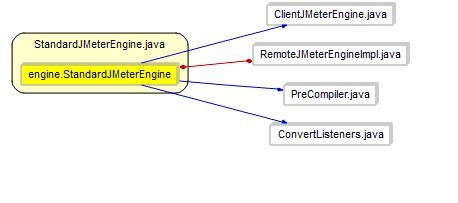
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.StandardJMeterEngine (Класс 6)**



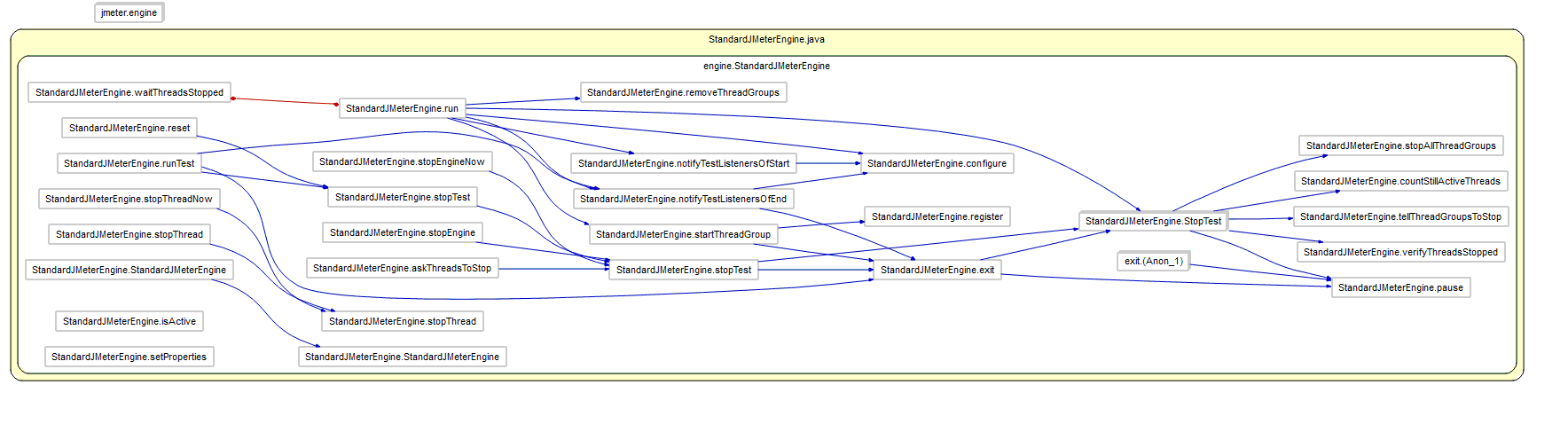
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.StandardJMeterEngine



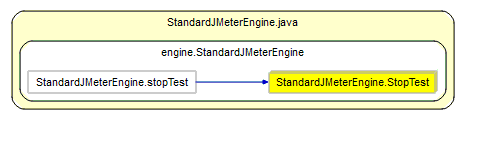
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.StandardJMeterEngine и какие классы вызывает сам класс



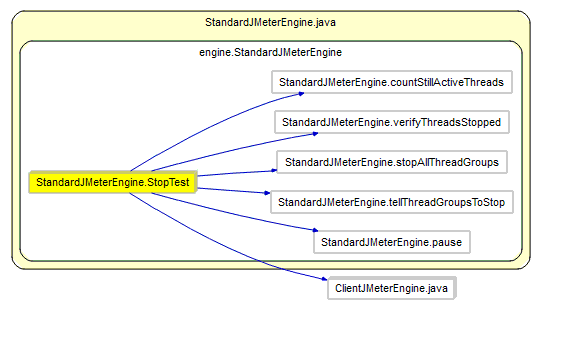
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.StandardJMeterEngine



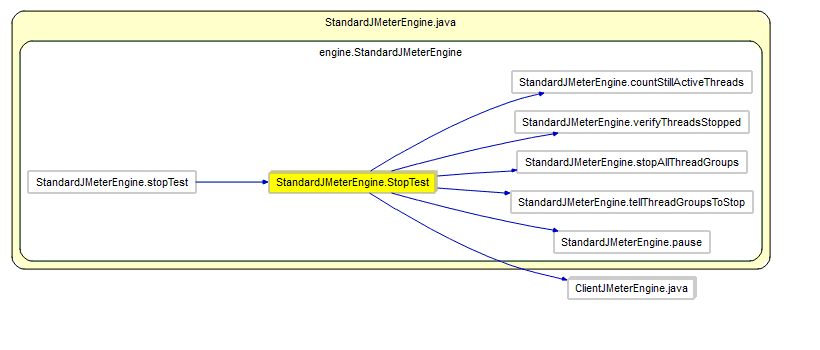
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.StandardJMeterEngine.StopTest (Подкласс 6)**



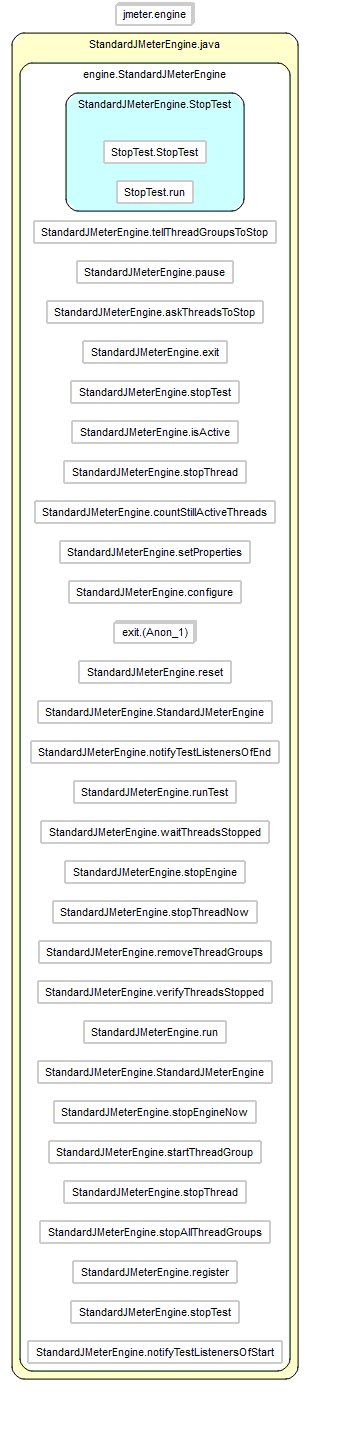
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.StandardJMeterEngine.StopTest



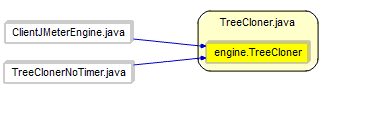
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.StandardJMeterEngine.StopTest и какие классы вызывает сам класс



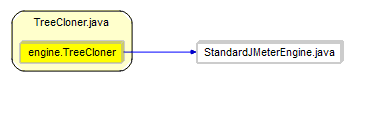
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.StandardJMeterEngine.StopTest



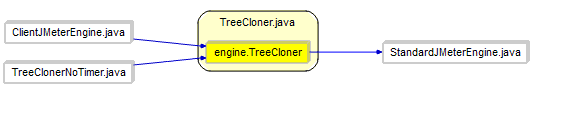
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.TreeCloner (Класс 7)**



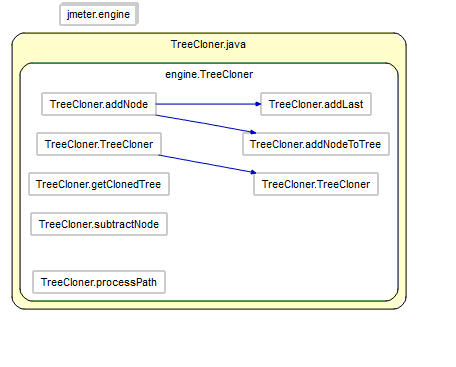
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.TreeCloner



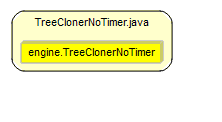
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.TreeCloner и какие классы вызывает сам класс



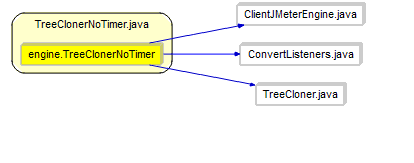
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.TreeCloner



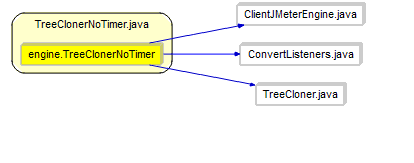
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.TreeClonerNoTimer (Класс 8)**



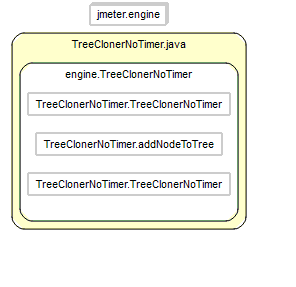
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.TreeClonerNoTimer



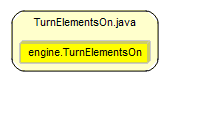
Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.TreeClonerNoTimer и какие классы вызывает сам класс



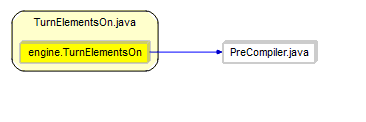
Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.TreeClonerNoTimer



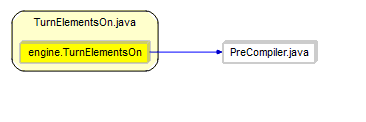
Разберемся, из каких классов вызывается класс **org.apache.jmeter.engine.TurnElementsOn (Класс 9)**



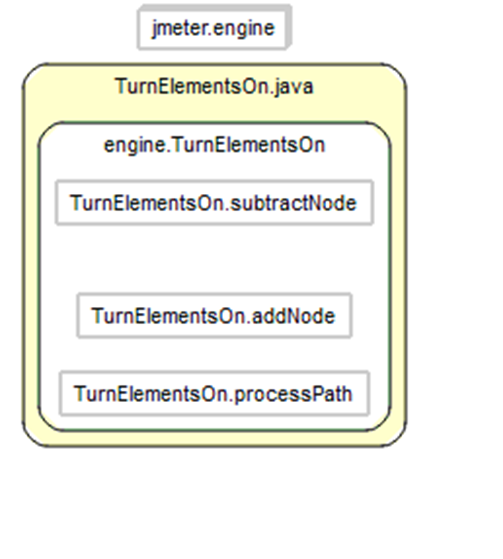
Узнаем, какие классы вызываются из класса org.apache.jmeter.engine.TurnElementsOn



Рассмотрим, из каких классов вызывается класс org.apache.jmeter.engine.TurnElementsOn и какие классы вызывает сам класс



Узнаем о зависимостях между методами внутри класса org.apache.jmeter.engine.TurnElementsOn



1. **МАТРИЦА ВЫЗОВОВ**

Построим матрицу вызовов между классами:

Класс 1 - org.apache.jmeter.engine.ClientMeterEngine

Класс 2 - org.apache.jmeter.engine.ConvertListeners

Класс 3 - org.apache.jmeter.engine.JMeterEngineException

Класс 4 - org.apache.jmeter.engine.PreCompiler

Класс 5 - org.apache.jmeter.engine.RemoteJMeterEngineImpl

Класс 6 - org.apache.jmeter.engine.StandardJMeterEngine

Класс 7 - org.apache.jmeter.engine.TreeCloner

Класс 8 - org.apache.jmeter.engine.TreeClonerNoTimer

Класс 9 - org.apache.jmeter.engine.TurnElementsOn

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Вызывающие классы | | | | | | | | |
| Класс 1 | Класс 2 | Класс 3 | Класс 4 | Класс 5 | Класс 6 | Класс 7 | Класс 8 | Класс 9 |
| Вызываемые классы | Класс 1 |  | + |  | + | + | + |  | + |  |
| Класс 2 |  |  |  |  | + | + |  | + |  |
| Класс 3 | + |  |  |  | + |  |  |  |  |
| Класс 4 | + |  |  |  |  | + |  |  | + |
| Класс 5 | + |  |  |  |  | + |  |  |  |
| Класс 6 |  |  |  |  | + |  | + |  |  |
| Класс 7 | + |  |  |  |  |  |  | + |  |
| Класс 8 |  |  |  |  |  |  |  |  |  |
| Класс 9 |  |  |  |  |  |  |  |  |  |

Исходя из матрицы вызовов между классами, можно сделать вывод, что точками входа в проект будут классы 8 и 9.

От точек входа построим все возможные статические маршруты:

8->1->3

8->1->4->1

8->1->5->1

8->1->5->2->1

8->1->5->3

8->1->5->6->1

8->1->5->6->2->1

8->1->5->6->4->1

8->1->5->6->5

8->2->1->3

8->2->1->4->1

8->2->1->5->1

8->2->1->5->2

8->2->1->5->3

8->2->1->6->1

8->2->1->6->2

8->2->1->6->4->1

8->2->1->6->5->1

8->2->1->6->5->2

8->2->1->6->5->3

8->2->1->6->5->6

8->7->6->1->3

8->7->6->1->4->1

8->7->6->1->5->1

8->7->6->1->5->2->1

8->7->6->1->5->3

8->7->6->1->5->6

8->7->6->1->7

8->7->6->2->1->3

8->7->6->2->1->4->2

8->7->6->2->1->5->1

8->7->6->2->1->5->2

8->7->6->2->1->5->3

8->7->6->2->1->5->6

9->4->1->3

9->4->1->4

9->4->1->5->1

9->4->1->5->2->1

9->4->1->5->3

9->4->1->5->6->1

9->4->1->5->6->2->1

9->4->1->5->6->4

9->4->1->5->6->5

1. **ВЫВОД**

Understand от Sci Tools - это инструмент статического анализа для поддержания, измерения и анализа критических или крупных баз кода.

Доступ через упрощенный информационный браузере обеспечивает всесторонний обзор программного обеспечения в быстрой и легкой форме. Быстрое проникновение в базы кода включено через широкий спектр показателей, настраиваемые отчеты, графики, набор функций анализа зависимости и точных средств поиска. В сочетании с настраиваемым, полнофункциональным программным интерфейсом, он является лучшим редактором программирования для поддержания и понимания кода[4].

1. **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**
   1. https://ru.wikipedia.org/wiki/%CC%E5%F2%F0%E8%EA%E0\_%EF%F0%EE%E3%F0%E0%EC%EC%ED%EE%E3%EE\_%EE%E1%E5%F1%EF%E5%F7%E5%ED%E8%FF
   2. http://dic.academic.ru/dic.nsf/ruwiki/176953
   3. https://scitools.com/feature-category/metrics-reports/
   4. http://www.emenda.eu/ru/products/understand

Размещено на Allbest.ru