МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)
Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

ПРАКТИЧЕСКАЯ РАБОТА №5

по дисциплине
«Тестирование и верификация программного обеспечения»
Тема: **«Статические и динамические анализаторы»**

Студенты группы ИКБО-20-21
Фомичев Р.А.
Сидоров С.Д.
Опришко В.Д.

_____
(подпись студента)

Принял руководитель работы Овчинникова М.А.

_____
(подпись руководителя)

Практические работы выполнены «___»_____2023 г.

Зачтено «___»_____2023 г.

Москва 2023

Содержание

1. Введение

1.1 Задание для выполнения

Проанализировать учебный код статическим анализатором.

Проанализировать учебный код динамическим анализатором.

1.2 Материальная часть

1.2.1 Статический анализ исходного кода

Статический анализ исходного кода программного обеспечения (ПО) представляет собой процесс анализа кода без его фактического выполнения. Этот вид анализа помогает обнаруживать потенциальные ошибки и проблемы в коде, такие как ошибки программирования, несоответствие стандартам кодирования, потенциальные уязвимости безопасности и другие проблемы производительности. Он выполняется с помощью специальных инструментов, известных как статические анализаторы, которые сканируют исходный код программы, исследуя его синтаксис, структуру, поток управления, зависимости между модулями и другие характеристики.

1.2.2. Динамический анализ исходного кода

Динамический анализ исходного кода программного обеспечения (ПО) представляет собой процесс анализа программного кода во время его выполнения. В отличие от статического анализа, который анализирует код без его выполнения, динамический анализ осуществляет оценку поведения программы в реальном времени. Этот процесс может включать в себя тестирование, отладку и профилирование программного обеспечения.

2. Инструменты

2.1. Используемое ПО

Для проведения тестирования программных продуктов использовались инструменты SpotBugs, PMD, Sonar Java для статического анализа и Java VisulIVM, Jprofiler и Cobertura для динамического анализа.

3. Выполнение задания

3.1. Статический анализ

3.1.1. PMD

PMD – статический анализатор исходного кода с открытым исходным кодом, который сообщает о проблемах, обнаруженных в коде приложения.

Результат работы анализатора представлен на рисунках 1-3.

PMD report

Problems found

#	File	Line	Problem
1	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\Equipment.java	5	Each class should declare at least one constructor
1	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\Main.java	3	Avoid short class names like Main
1	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	11	Parameter 'equipment' is not assigned and could be declared final
2	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	11	Parameter 'userInput' is not assigned and could be declared final
3	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	14	A method should have only one exit point, and that should be the last statement in the method
4	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	18	Parameter 'userInput' is not assigned and could be declared final
5	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	20	A method should have only one exit point, and that should be the last statement in the method
6	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	20	This statement should have braces
7	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	23	Parameter 'userInput' is not assigned and could be declared final
8	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	24	Consider simply returning the value vs storing it in local variable 'lowerLetter'
9	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\CheckEnteryString.java	24	Local variable 'lowerLetter' could be declared final
1	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\PerformUserInput.java	11	Parameter 'equipment' is not assigned and could be declared final
2	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\PerformUserInput.java	11	Parameter 'userInput' is not assigned and could be declared final
3	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\PerformUserInput.java	14	A method should have only one exit point, and that should be the last statement in the method
4	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\PerformUserInput.java	20	Parameter 'equipment' is not assigned and could be declared final
5	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\PerformUserInput.java	20	Parameter 'userInput' is not assigned and could be declared final
6	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\PerformUserInput.java	29	Parameter 'equipment' is not assigned and could be declared final
7	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\bdd\PerformUserInput.java	29	Parameter 'userInput' is not assigned and could be declared final
1	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	11	Parameter 'equipment' is not assigned and could be declared final
2	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	11	Parameter 'userInput' is not assigned and could be declared final
3	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	13	A method should have only one exit point, and that should be the last statement in the method
4	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	17	Parameter 'userInput' is not assigned and could be declared final
5	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	19	A method should have only one exit point, and that should be the last statement in the method
6	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	19	This statement should have braces
7	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	22	Parameter 'userInput' is not assigned and could be declared final
8	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	23	Consider simply returning the value vs storing it in local variable 'lowerLetter'
9	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\CheckEnteryString.java	23	Local variable 'lowerLetter' could be declared final
1	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\PerformUserInput.java	12	Parameter 'equipment' is not assigned and could be declared final
2	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\PerformUserInput.java	12	Parameter 'userInput' is not assigned and could be declared final
3	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\PerformUserInput.java	15	A method should have only one exit point, and that should be the last statement in the method
4	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\PerformUserInput.java	21	Parameter 'equipment' is not assigned and could be declared final
5	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\PerformUserInput.java	21	Parameter 'userInput' is not assigned and could be declared final
6	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\PerformUserInput.java	30	Parameter 'equipment' is not assigned and could be declared final
7	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\main\java\org\example\tdd\PerformUserInput.java	30	Parameter 'userInput' is not assigned and could be declared final
1	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\test\java\test\EquipmentTest.java	5	Unused static import 'org.junit.jupiter.api.Assertions.*'
2	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\test\java\test\EquipmentTest.java	7	Each class should declare at least one constructor
1	C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src\test\java\test\bdd\CheckEnteryStringStepDef.java	3	Unused import 'io.cucumber.java.bs.A'

Рисунок 1 – Вывод проблем

```
C:\Users\sidor>pmd.bat check -d "C:\Users\sidor\OneDrive\Рабочий стол\TiVPOMIREA3\src" -R "C:\Users\sidor\OneDrive\Рабо
ий стол\TiVPOMIREA3\pmd-ruleset.xml" -f html > results.html
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.reflection.CachedClass (file:/C:/pmdlib/lib/groovy-2.4.21.jar
 to method java.lang.Object.finalize()
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.reflection.CachedClass
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[main] INFO net.sourceforge.pmd.cli - Log level is at INFO
[main] WARN net.sourceforge.pmd.cli - Progressbar rendering conflicts with reporting to STDOUT. No progressbar will be
hown. Try running with argument -r <file> to output the report to a file instead.
[main] WARN net.sourceforge.pmd.cli - This analysis could be faster, please consider using Incremental Analysis: https:
/docs.pmd-code.org/pmd-doc-7.0.0-rc4/pmd_userdocs_incremental_analysis.html
```

Рисунок 2 – Запуск

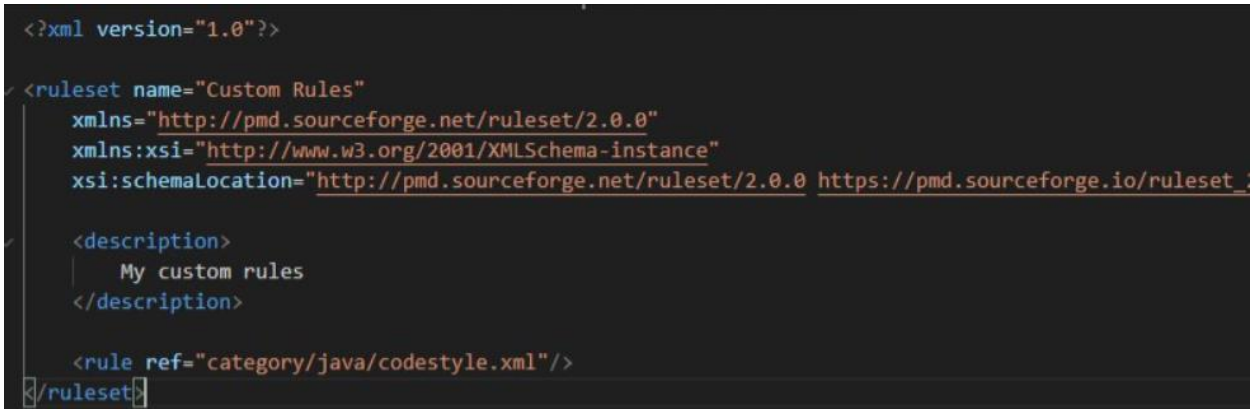

Рисунок 3 – Файл конфигурации

3.1.2. SpotBug

Spotbug – это инструмент, который позволяет исследовать код для поиска возможных проблем. Он производит статичный анализ, чтобы найти определенные антипаттерны, которые могут вызвать проблемы(например, низкую производительность).

Результат работы анализатора представлен на рисунках 4-6.

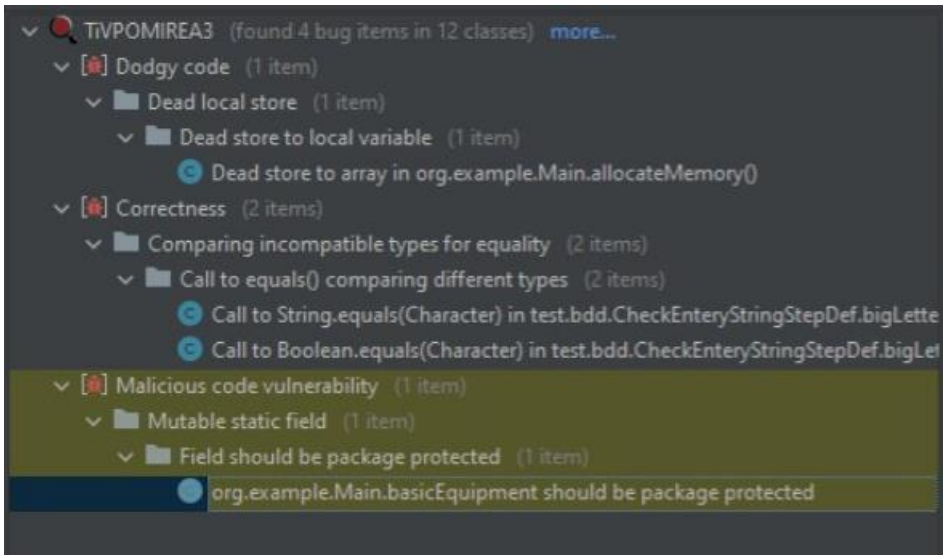

Рисунок 4 – Список проблем

Рисунок 5 – Превью проблемы

Рисунок 6 – Описание проблемы

3.1.3. SonarJava

SonarJava – это статический анализатор кода для Java от SonarSource. Этот анализатор может работать бок о бок со встроенным анализатором IntelliJ IDEA.

Результат работы анализатора представлен на рисунках 7, 8.

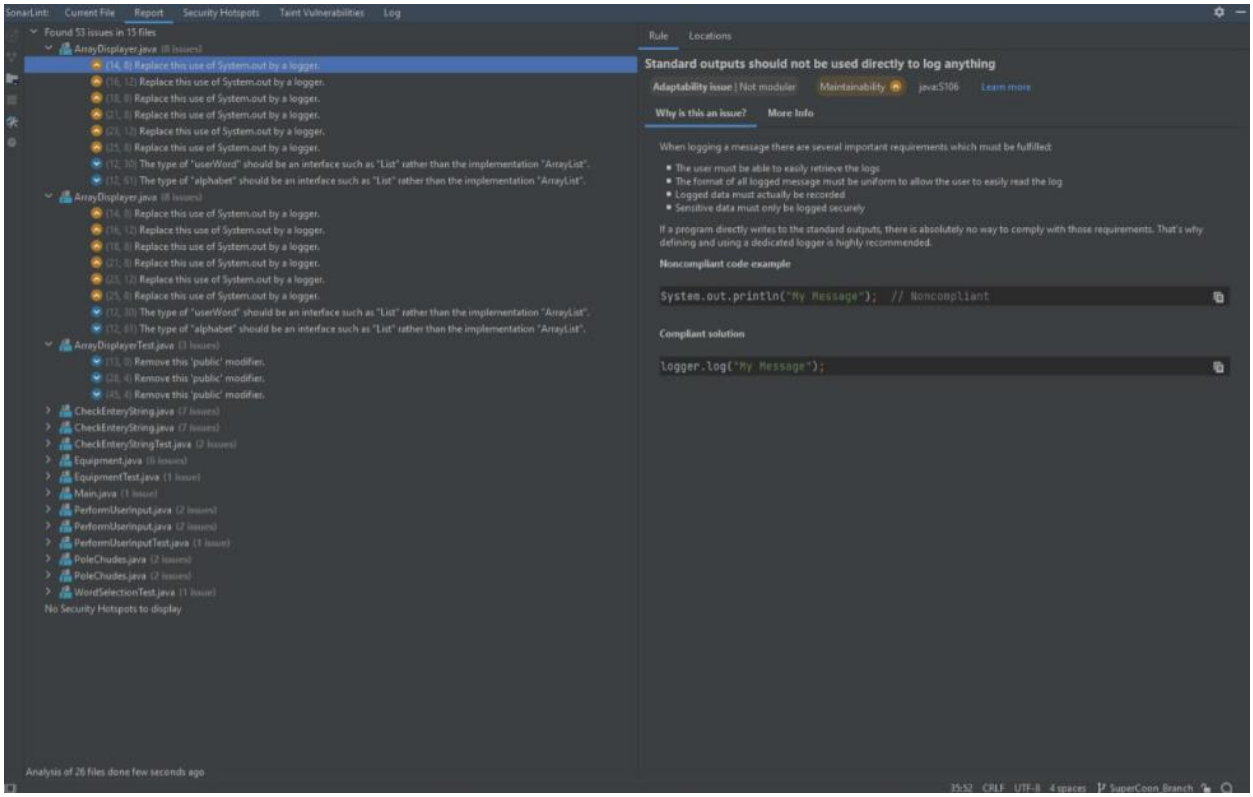

Рисунок 7

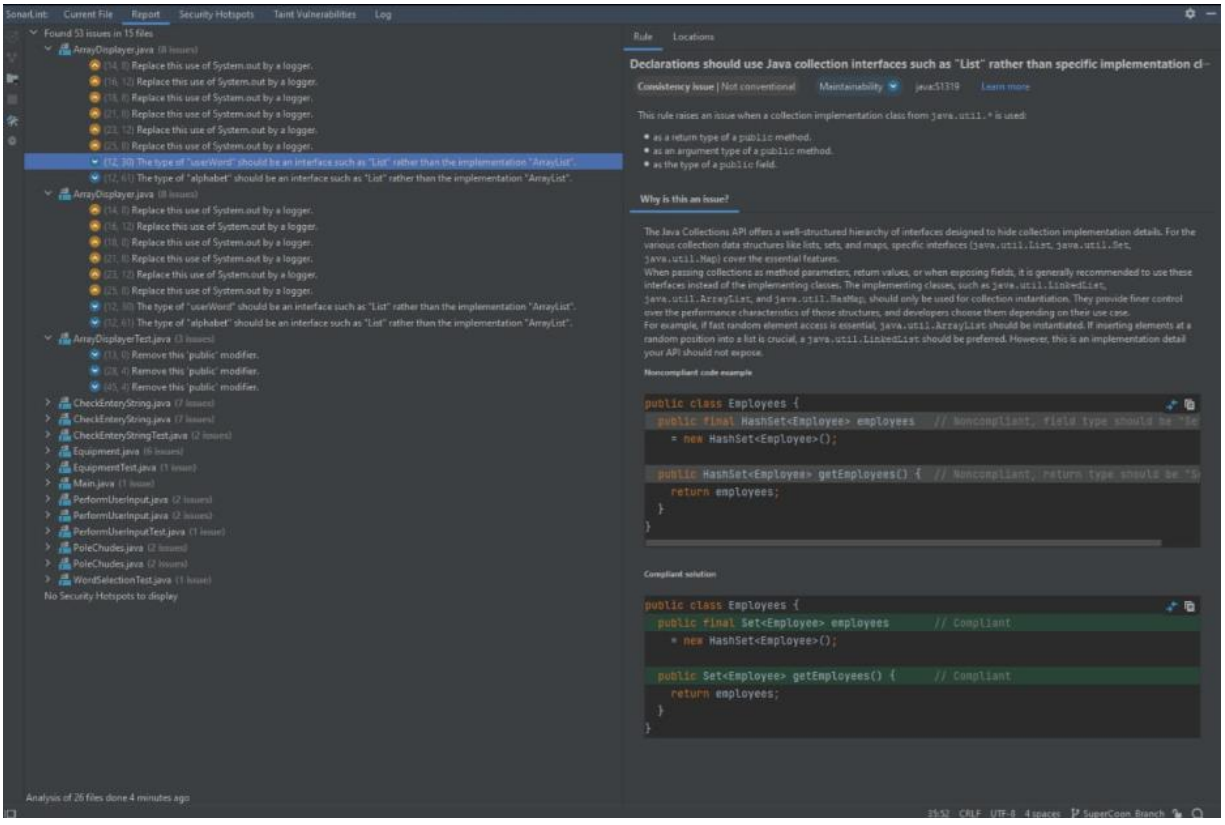

Рисунок 8

3.2. Динамический анализ

3.2.1. JProfiler

JProfiler – комплексный профилировщик Java. Интуитивный пользовательский интерфейс Jprofiler поможет устранить узкие места производительности, точно определить утечки памяти и понять проблемы многопоточности.

Результат работы анализатора представлен на рисунке 9.

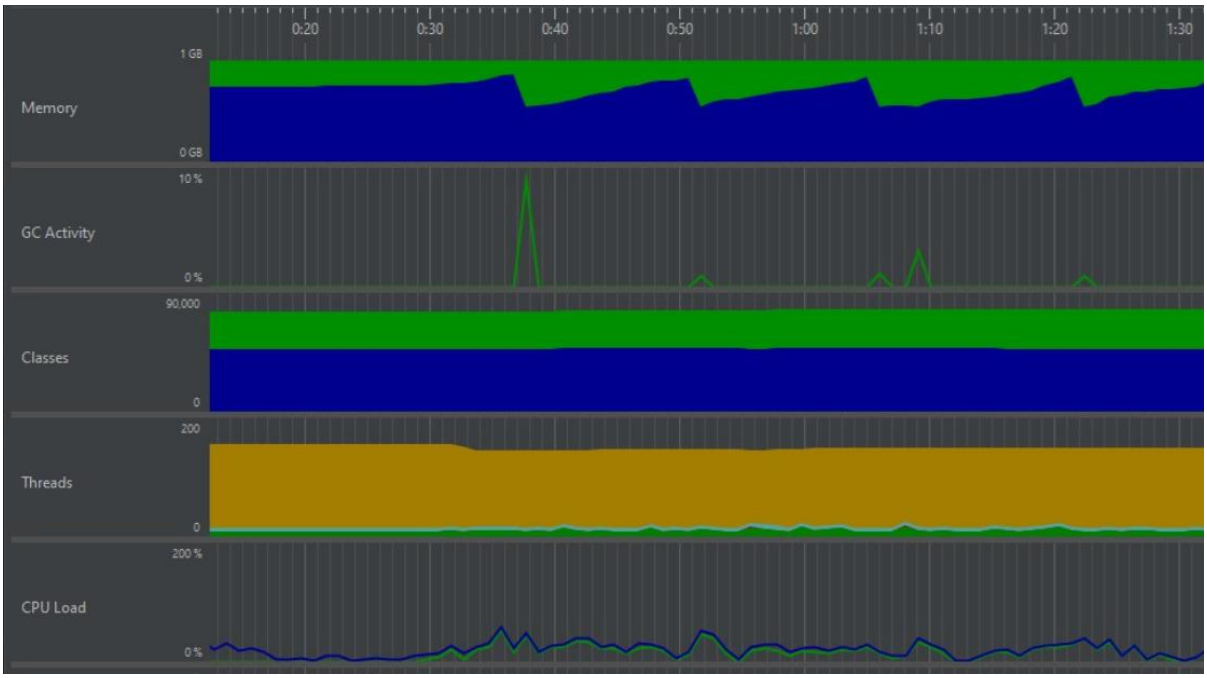

Рисунок 9 – Результат работы

3.2.2. Java VisualVM

Java VisualVM – это инструмент, предоставляющий визуальный интерфейс для просмотра подробной информации о приложениях Java во время их работы на виртуальной машине.

Результат работы анализатора представлен на рисунке 10.

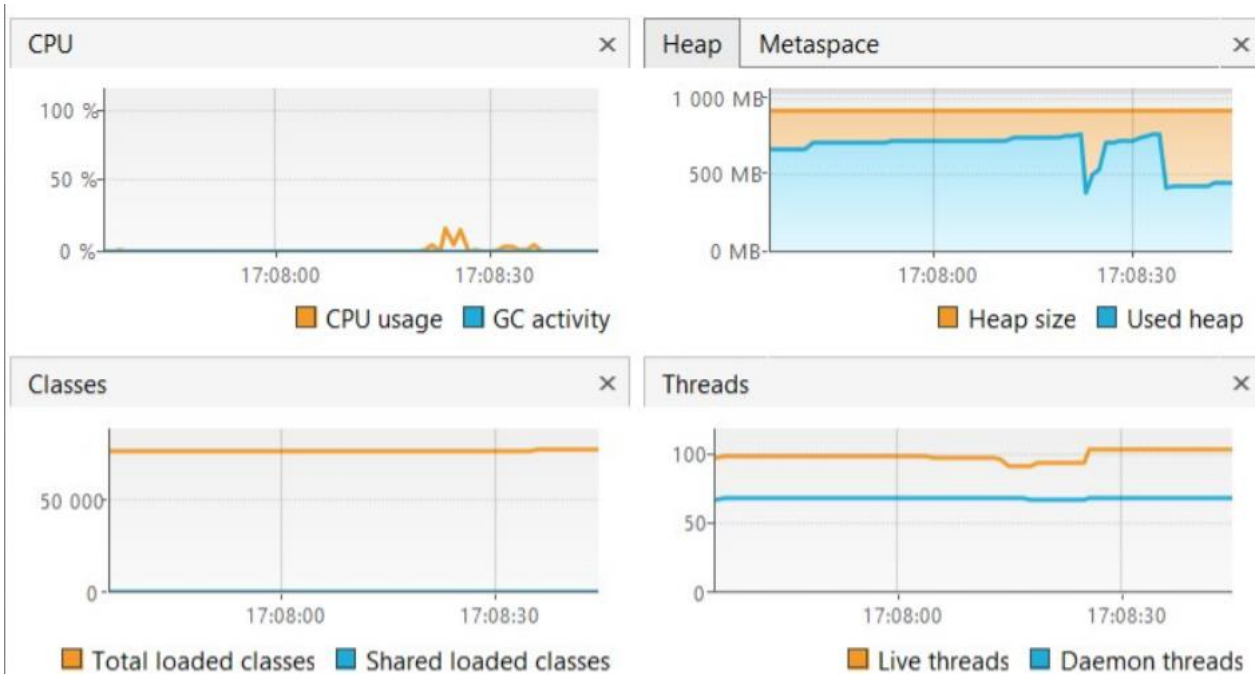

Рисунок 10 – Результат работы

3.2.3. Cobertura

Cobertura – плагин для Maven, позволяющий получать отчет покрытия исходного Java кода проекта тестами.

Результат работы анализатора представлен на рисунке 11.

File	Lines	Line Coverage	Functions	Function Coverage	Branches	Branch Coverage
– All Files	586/625	93.8%	40/41	97.6%	105/118	89.0%
– packages	586/625	93.8%	40/41	97.6%	105/118	89.0%
– app	200/207	96.6%	11/12	91.7%	28/34	82.4%
– src	200/207	96.6%	11/12	91.7%	28/34	82.4%
+ components	184/189	97.4%	9/9	100.0%	25/28	89.3%
+ utils	16/18	88.9%	2/3	66.7%	3/6	50.0%
+ components	259/290	89.3%	20/20	100.0%	43/48	89.6%
– core	127/128	99.2%	9/9	100.0%	34/36	94.4%
index.js	7/7	100.0%	0/0	100.0%	0/0	100.0%
+ src	120/121	99.2%	9/9	100.0%	34/36	94.4%

Рисунок 11 – Результат работы

4. Выводы

Статические и динамические анализы программного продукта с использованием инструментов SpotBugs, PMD, SonarJava, Java VisualVM, JProfiler и Cobertura проведены успешно. При помощи данных технологий удалось существенно сократить время и автоматизировать анализ кода.

5. Список использованных источников

1. Статический анализ кода [Электронный ресурс] – https://www.jetbrains.com/ru-ru/resharper/features/code_analysis.html
2. Динамический анализ кода [Электронный ресурс] – https://habr.com/ru/companies/pvs-studio/articles/580196/
3. WebStrom – анализ кода [Электронный ресурс] – https://www.jetbrains.com/ru-ru/resharper/features/code_analysis.html
4. Использование статического и динамического анализа для повышения качества продукции и эффективности разработки [Электронный ресурс] – https://www.swd.ru/print.php3?pid=828

6. Дополнения и приложения