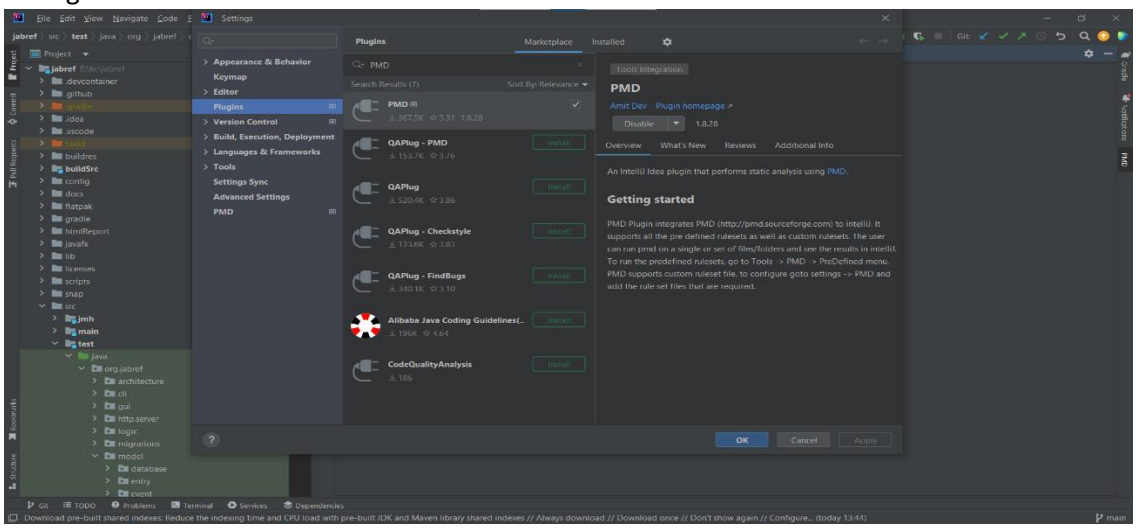
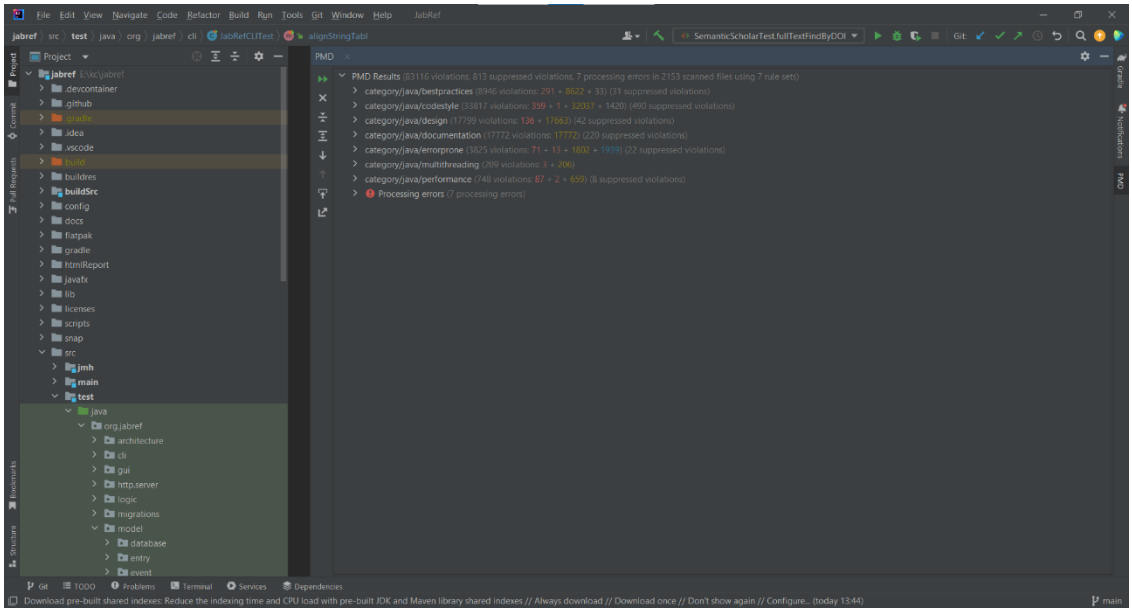
CSC 6110 : Software Engineering
Assignment 3
Fall 2023
Student name: Kanchan Chopde

Topic: Static Analysis

- Configured PMD:



- PMD report after run:



Analysis of 10 violations

a. File name and line numbers that include this violation.

1) BibEntryRelationsRepository.java , Line 40

b. What is the violation about?

It states to avoid throwing certain types of exceptions like raw RuntimeException, Throwable, Exception or Error , use subclassed exception or error instead.

c. Do you think it as an actual violation or a false alarm?

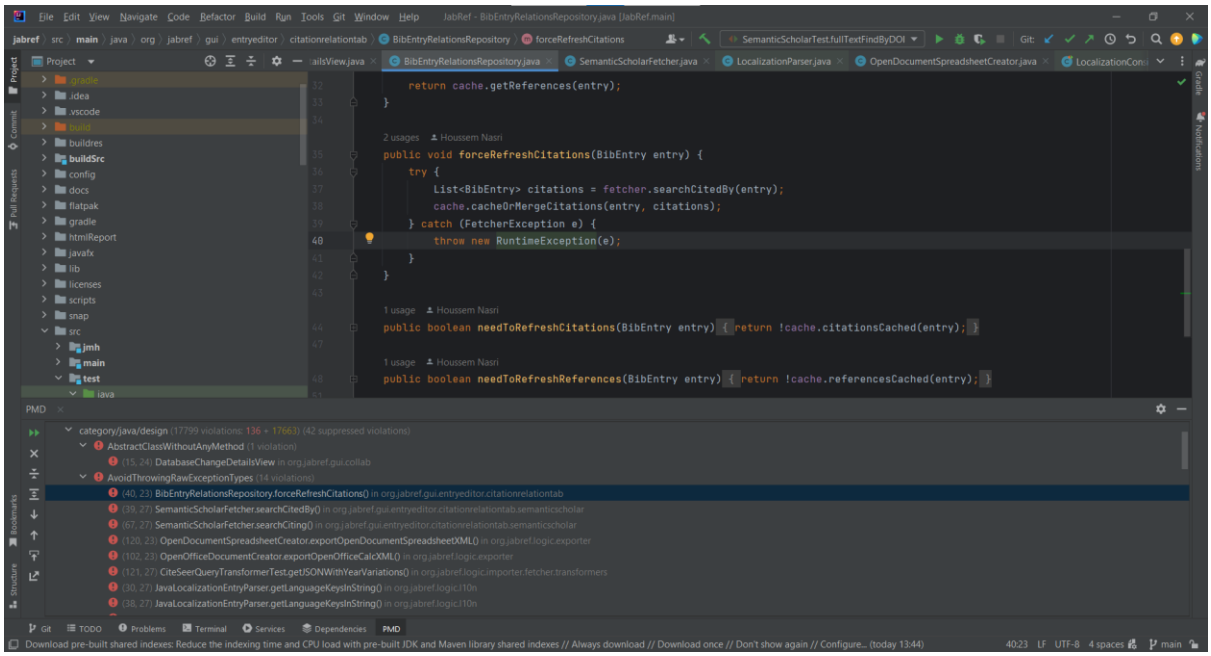
It is actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

Throwing a RuntimeException can lead to subtle errors, for example, a caller cannot examine the exception to determine why it was thrown and consequently cannot attempt recovery.

e. If true positive, how would fix it.

Declares a more specific exception class in the `throw` clause of the method declaration for the `ForceRefreshCitations()` method such as IOException or NullException.



a. File name and line numbers that include this violation.

2) FileFilterUtilsTests.java , Line 159

b. What is the violation about?

The violation is about improper class Naming convention.

c. Do you think it as an actual violation or a false alarm?

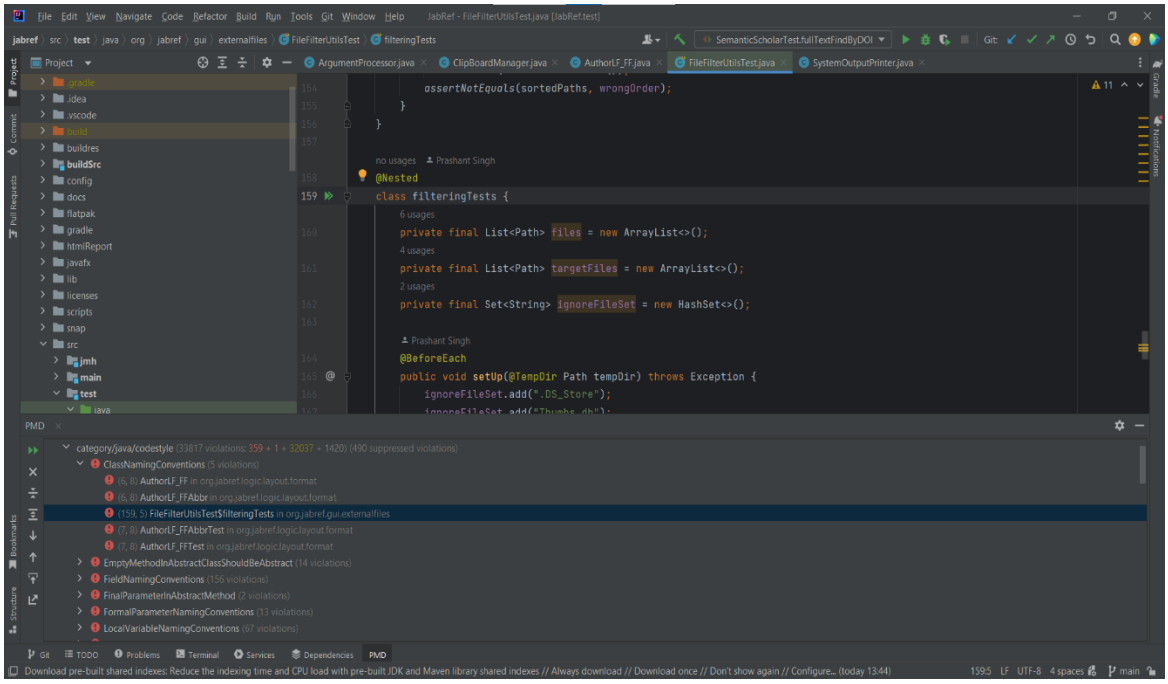
It is actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

It will cause maintenance issue.

e. If true positive, how would fix it.

Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).
(Ref: https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html)



a. File name and line numbers that include this violation.

3) ContextMenuAddable.java , Line 14

b. What is the violation about?

The violation is about avoiding usage of final parameter in abstract method.

c. Do you think it as an actual violation or a false alarm?

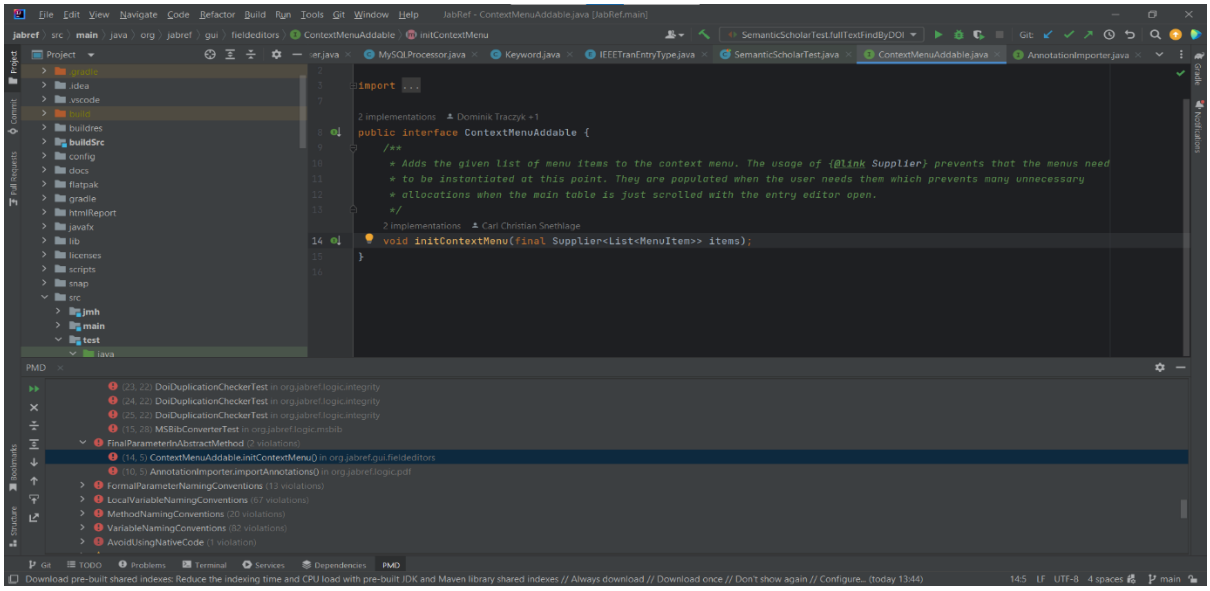
It is actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

It will not cause issue but it is codestyle violation.

e. If true positive, how would fix it.

To fix this issue, we can simply remove final keyword as it is unnecessary.
The purpose of making a parameter final is to indicate that the parameter should not be changed within the method. However, in an abstract method, there's no method body to make such changes. Abstract methods only provide a method signature, and the actual implementation is provided by concrete subclasses.



a. File name and line numbers that include this violation.

4) Sources.java, Line 4

b. What is the violation about?

Violation is about Field naming Convention not being followed.

c. Do you think it as an actual violation or a false alarm?

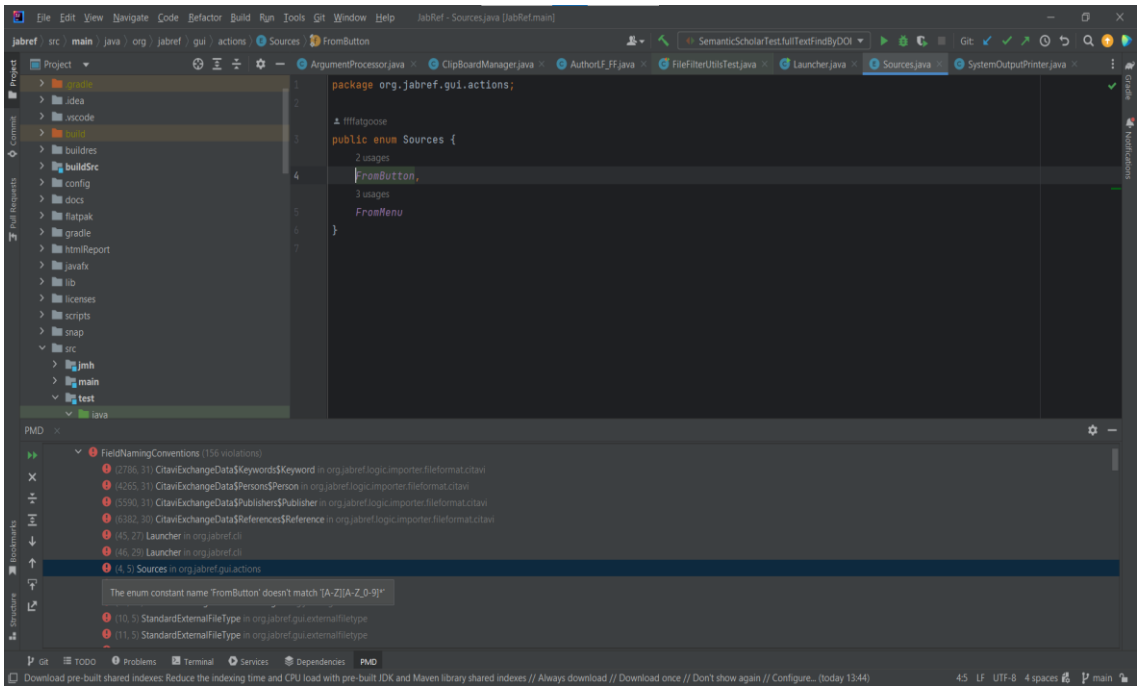
It is an actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

It can cause maintenance issue.

e. If true positive, how would fix it.

Use standard/preferred naming conventions. Because they are constants, the names of an enum type's fields are in uppercase letters.



a. File name and line numbers that include this violation.

5) AutoCompletionTextInputBinding.java , Line 133

b. What is the violation about?

It states to avoid reassignment of parameters.

c. Do you think it as an actual violation or a false alarm?

It is an actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

It can cause maintenance issue.

e. If true positive, how would fix it.

The the issue of reassigning the 'newText' parameter can be fixed by creating a local variable to store the modified value, instead of directly modifying the parameter.

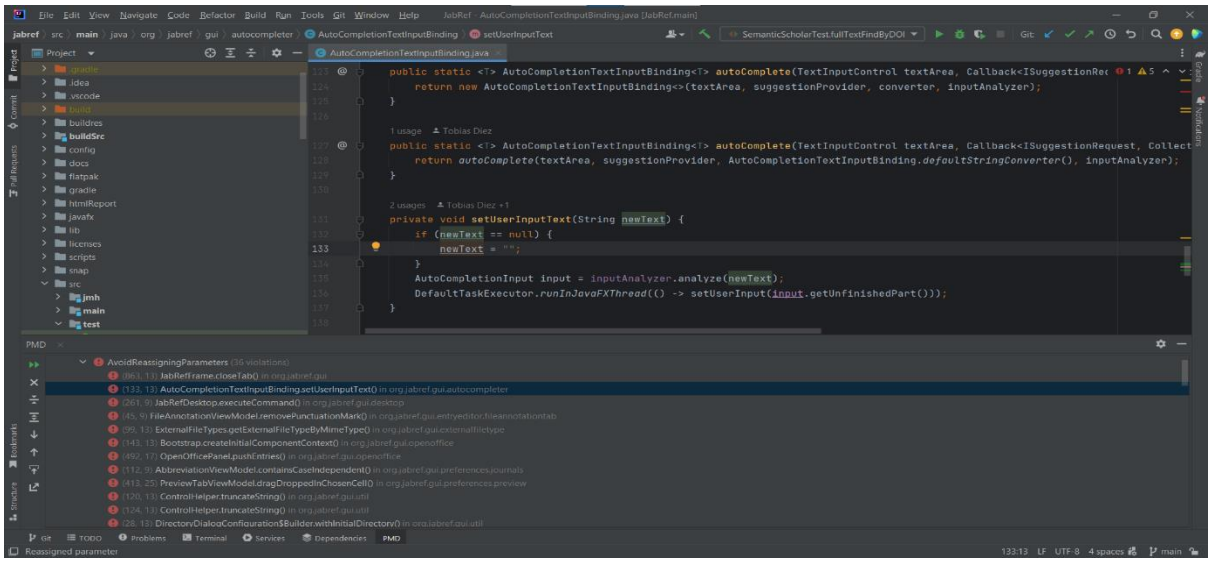
Eg. private void setUserInputText(String newText) {

String sanitizedText = (newText == null) ? "" : newText;

AutoCompletionInput input = inputAnalyzer.analyze(sanitizedText);

DefaultTaskExecutor.runInJavaFXThread(() -> setUserInput(input.getUnfinishedPart()));

}



a. File name and line numbers that include this violation.

6) AuthorListTest.java, Line 1097

b. What is the violation about?

It states to avoid string instantiation using new keyword.

c. Do you think it as an actual violation or a false alarm?

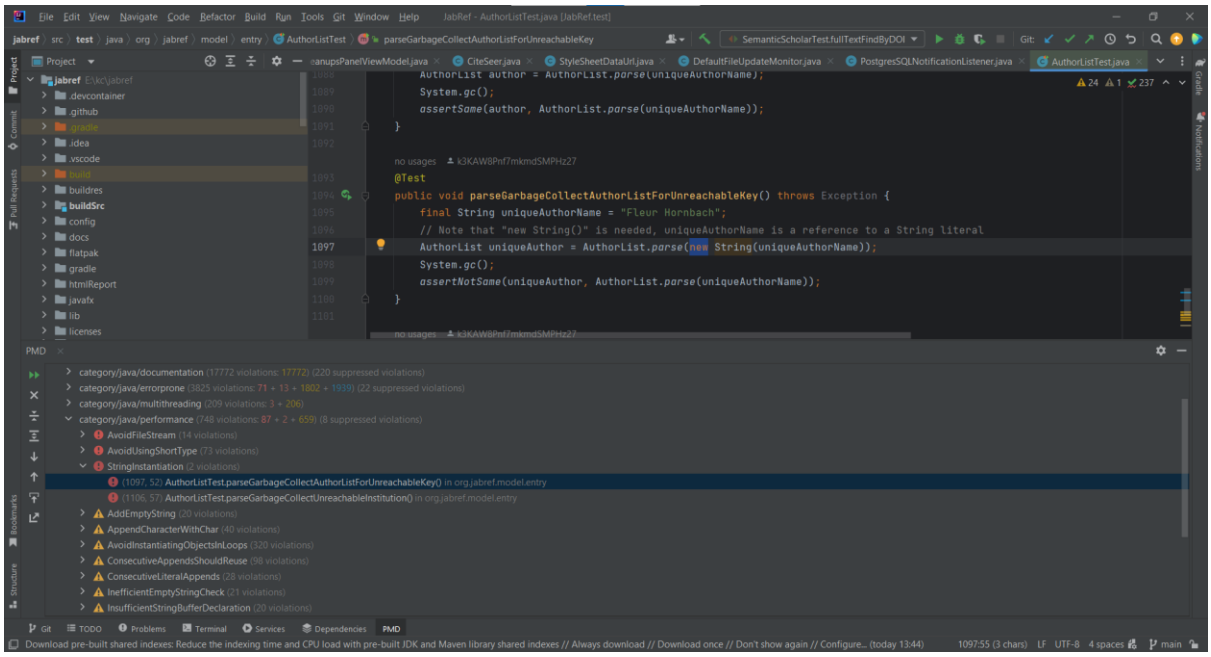
It is an actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

This bug can cause performance issue.

e. If true positive, how would fix it.

By using string literals. String literals are automatically interned and stored in the string pool, reducing the need for unnecessary object creation.



a. File name and line numbers that include this violation.

7) AuthorListTest.java, Line 1106

b. What is the violation about?

It states to avoid string instantiation using new keyword.

c. Do you think it as an actual violation or a false alarm?

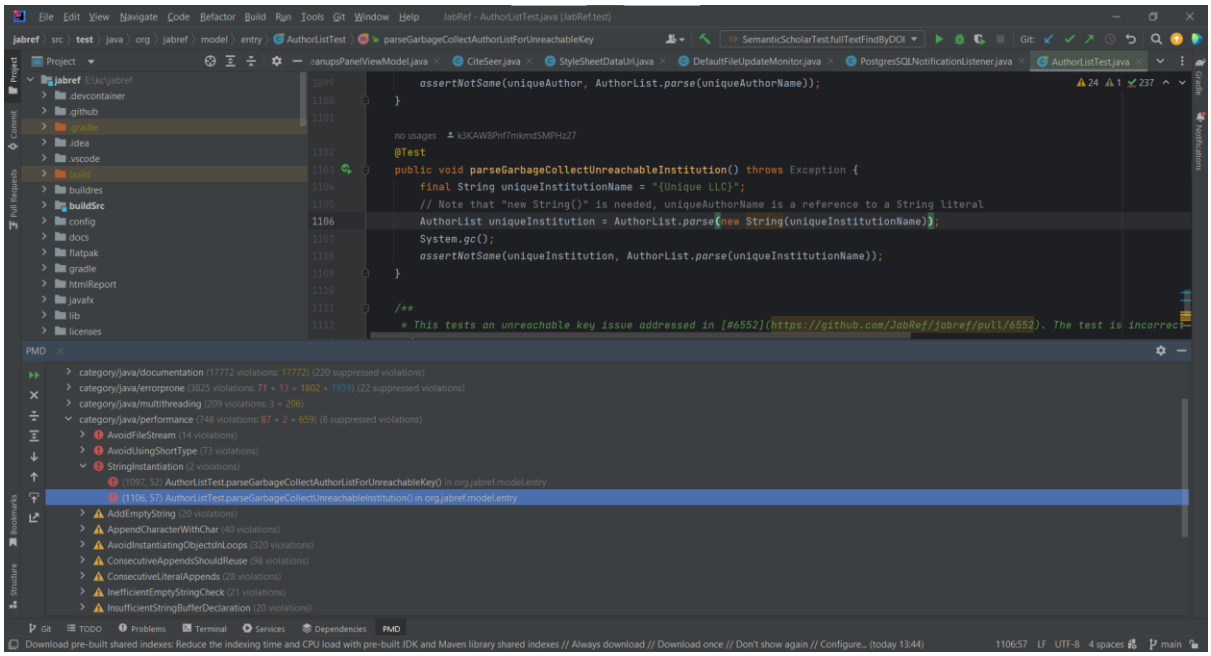
It is an actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

This bug can cause performance issue.

e. If true positive, how would fix it.

By using string literals. String literals are automatically interned and stored in the string pool, reducing the need for unnecessary object creation.



a. File name and line numbers that include this violation.

8) CitaviExchangeData.java , Line 4268

b. What is the violation about?

The violation is about avoid using short type.

c. Do you think it as an actual violation or a false alarm?

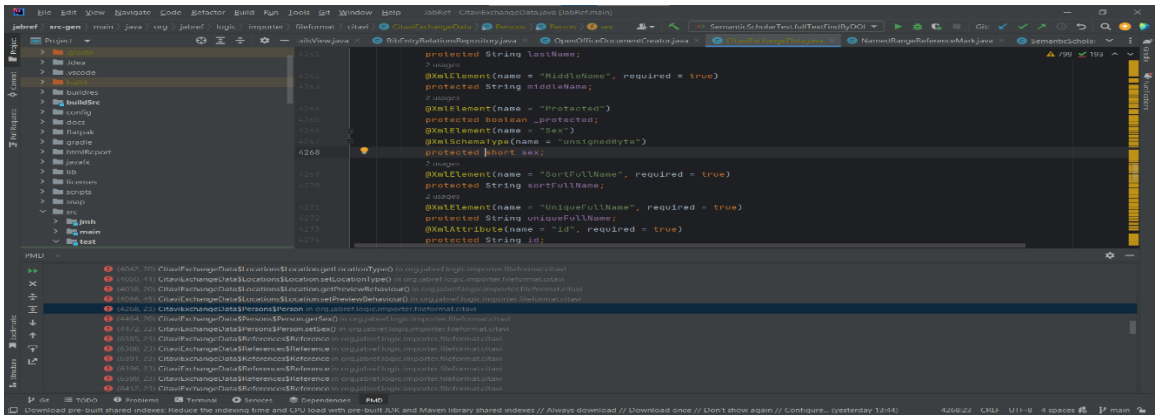
It is a actual violation.

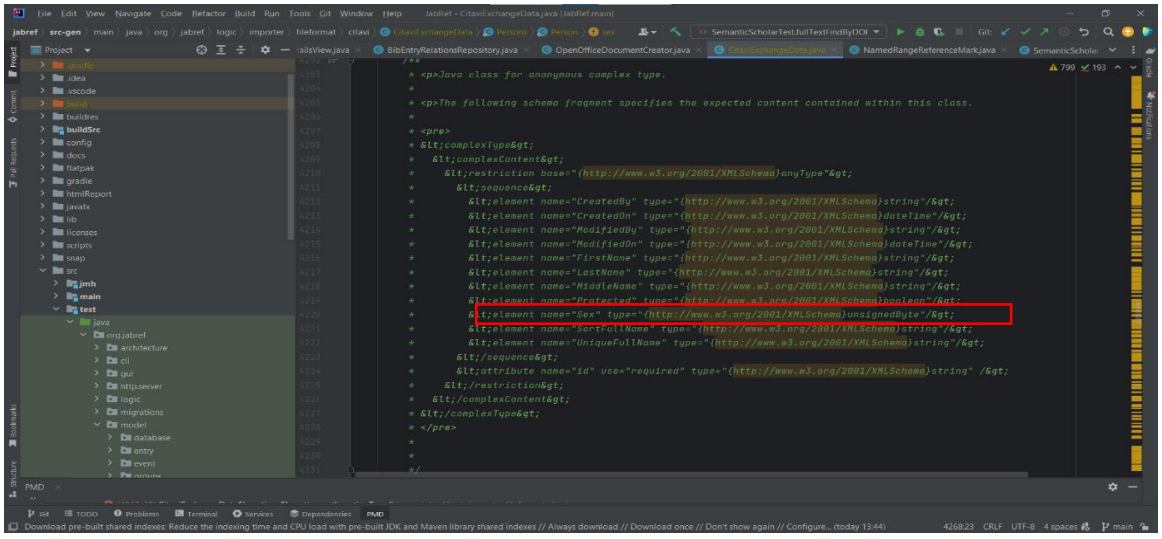
d. Does the violation cause bug /performance issue/ or maintenance issues?

It can cause bug. Usage of “short” for an "unsignedByte" field it can store values outside the intended range. The short data type in Java is a 16-bit signed integer, which means it can represent both positive and negative values. unsignedByte can have only non-negative values in range 0-255.

e. If true positive, how would fix it.

By using the “byte” data type for an "unsignedByte" field to satisfy the requirements.





a. File name and line numbers that include this violation.

9) BibStringDiffTest.java, Line 62

b. What is the violation about?

It is related to avoid using comparison of object to ‘null’ using equals() method.

c. Do you think it as an actual violation or a false alarm?

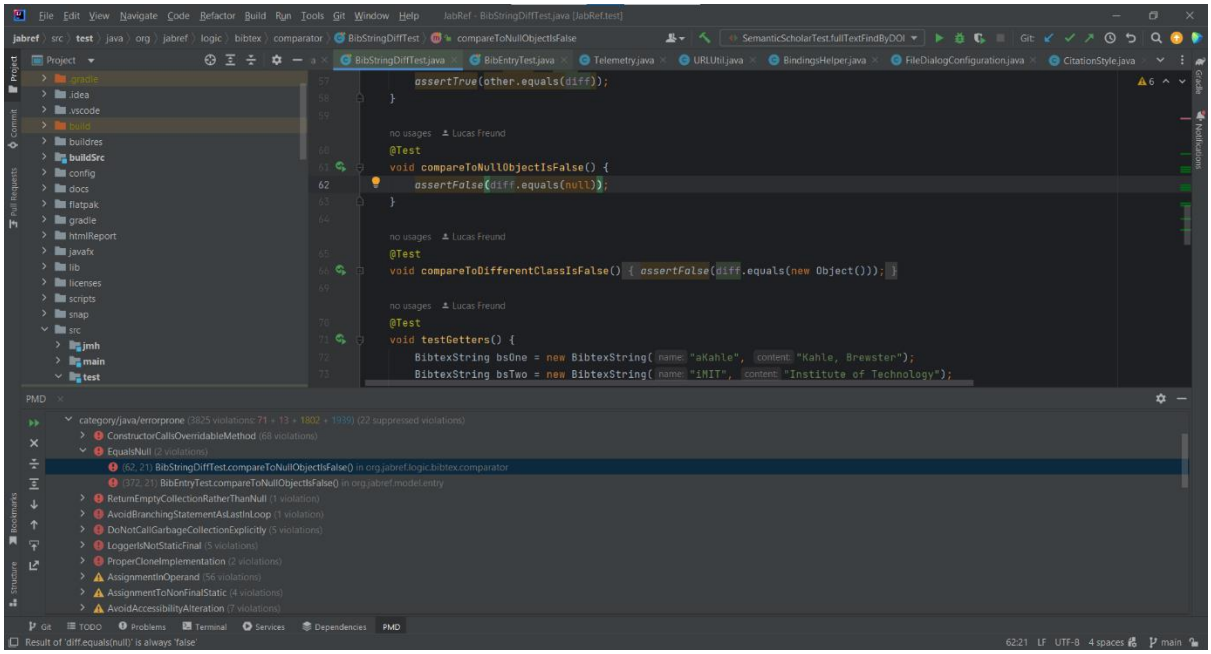
It is an actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

It can cause bug. It can cause an unexpected behaviour or errors in the code.

e. If true positive, how would fix it.

To perform comparison use equality operator “==” or not equal operator “!=”



a. File name and line numbers that include this violation.

10) CitaviExchangeData.java, Line 2786

b. What is the violation about?

This violation is about field naming convention not followed. Using an underscore "_protected " as a prefix is not a widely accepted naming convention in Java.

c. Do you think it as an actual violation or a false alarm?

It is an actual violation.

d. Does the violation cause bug /performance issue/ or maintenance issues?

It can cause maintenance issue.

e. If true positive, how would fix it.

While naming fields we can use lowercaseCamelCase instead to improve readablitiy and avoid confusion.

It can we re written as : protected boolean protectedField.