

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

SOFTWARE ENGINEERING II

Professor: Eng. Mónica Katuska Villavicencio Cabezas

Parallel: 2

Group: 9

Members:

- Dayse Joselyne Maroto Lema
- Ricardo Aurelio Rivera Guerra
- Yuleixi Katherine Garcia Anchundia

Start Date: June 10, 2021

End Date: June 10, 2021

Code Inspection - I Term 2021

Github link:

<https://github.com/dmaroto98/CodeInspection.git>

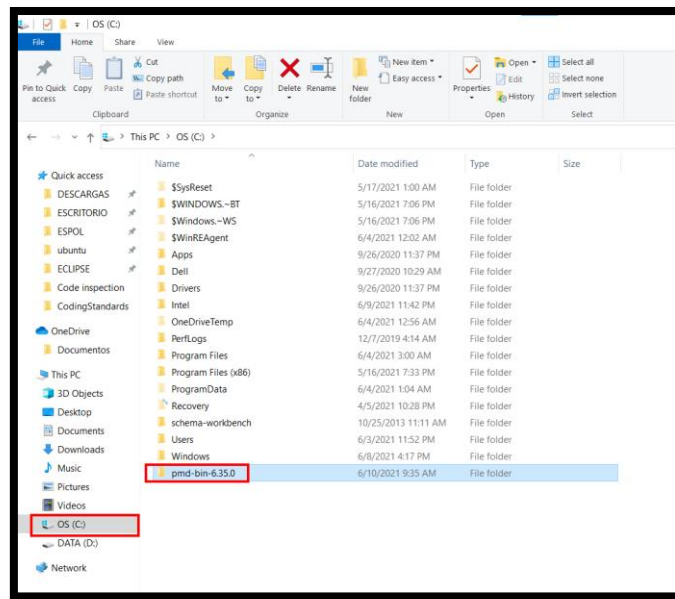
For this workshop you must have Maven, you can help yourself with this tutorial for its installation, part 1:

<https://github.com/leortyz/softwareEngineeringResources/wiki/Code-Inspection>

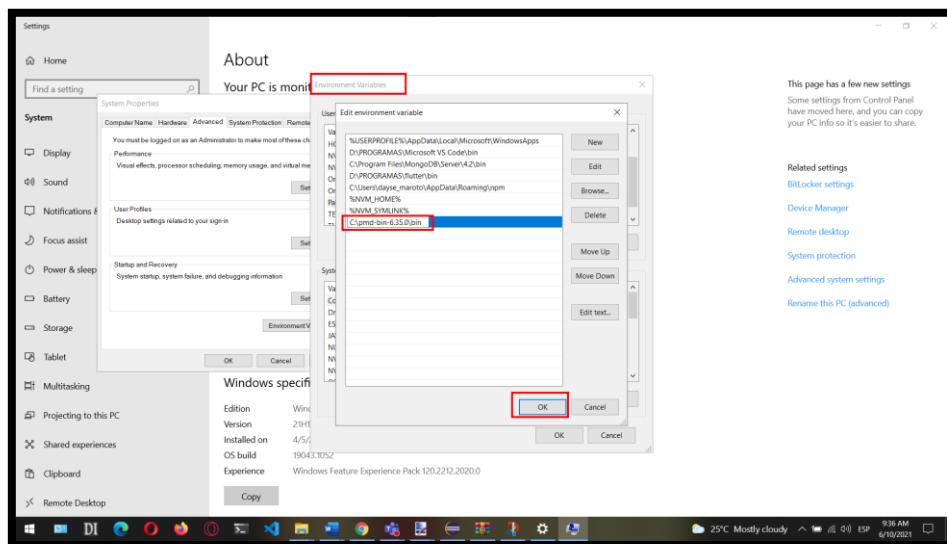
1) Results of activities

Part 2: Install the Eclipse PDM plug-in

1. Download [pmd-bin-6.35.0.zip](#)
2. Extract the zip-archive, e.g. to C:\pmd-bin-6.35.0



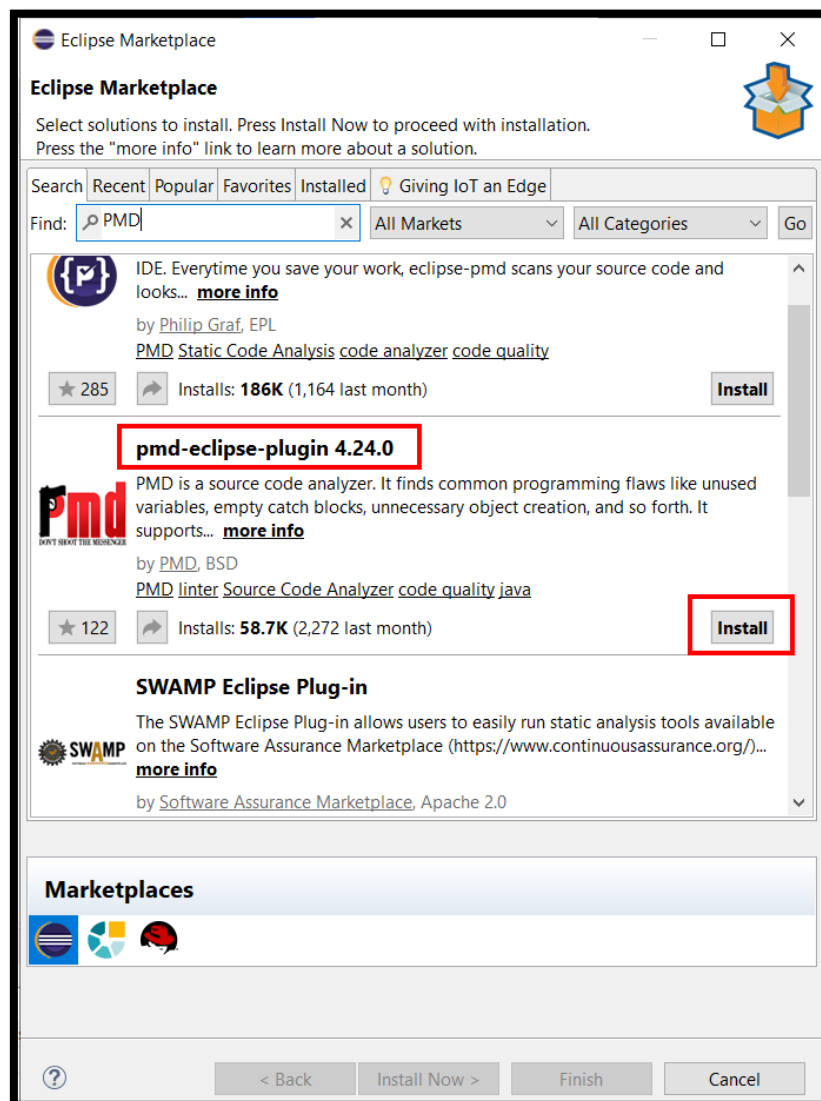
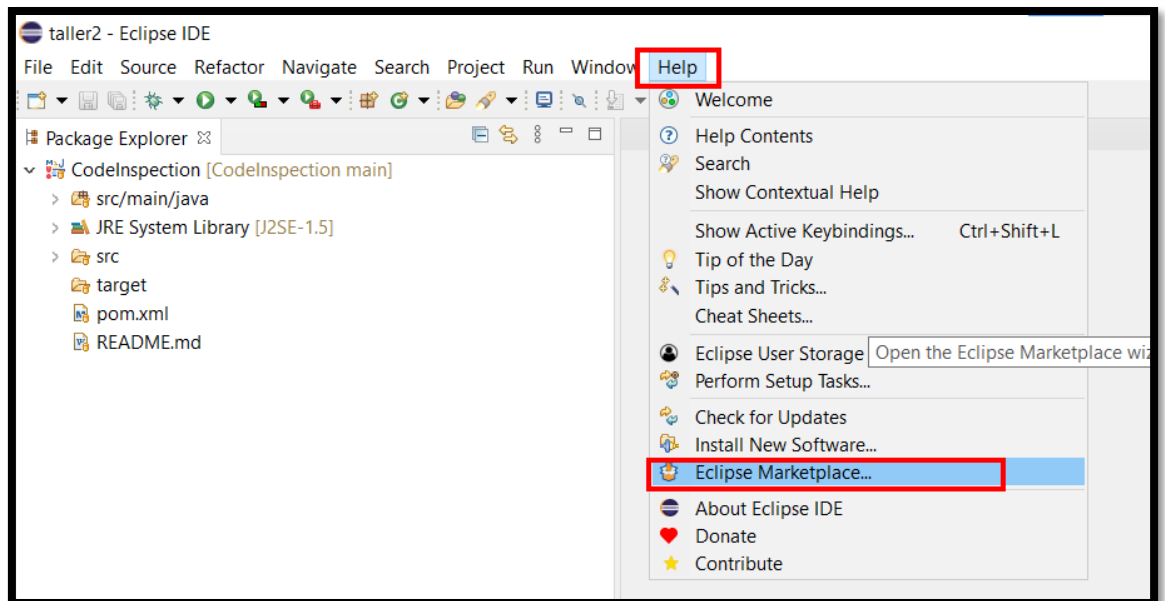
3. Add folder C:\pmd-bin-6.35.0\bin to PATH, either

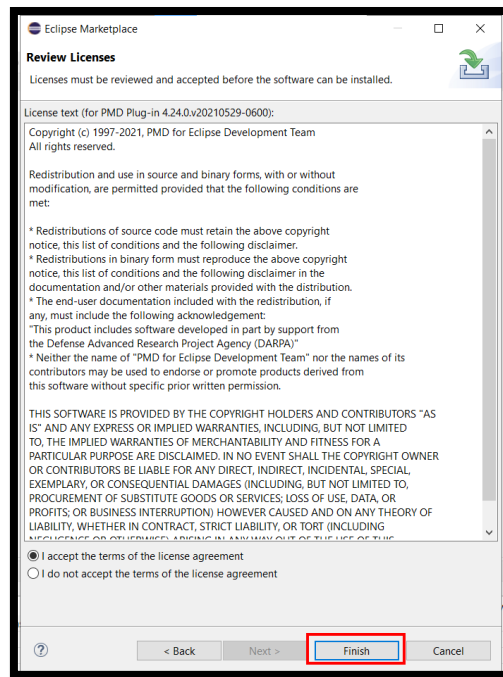


4. Execute at command line:

```
pmd.bat -d c:\src -R rulesets/java/quickstart.xml -f text
```

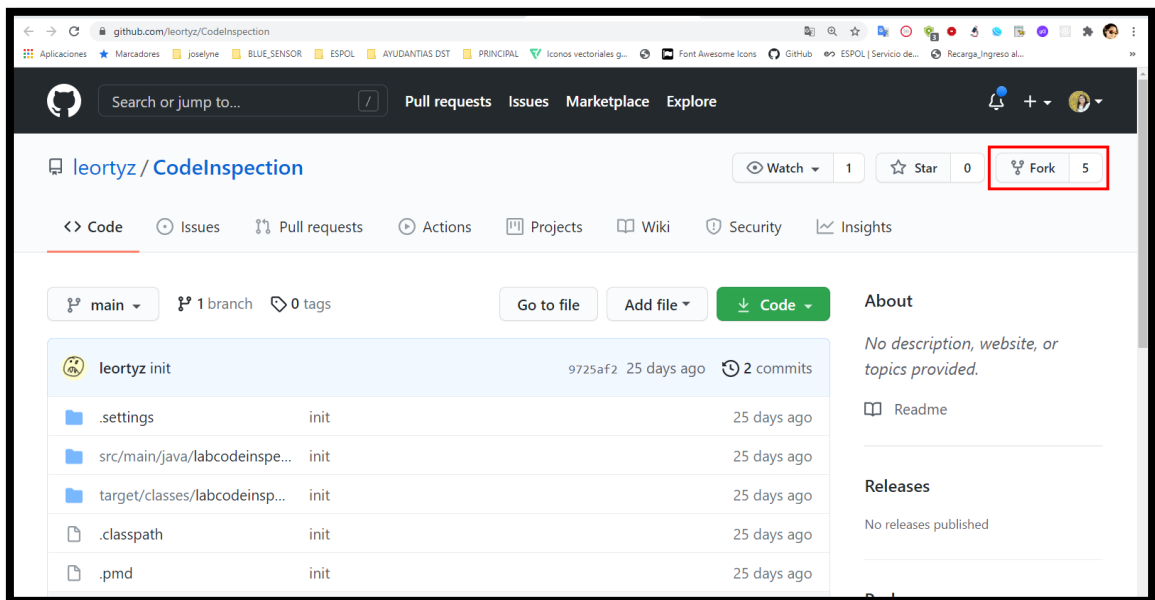
or download it from the eclipse marketplace.

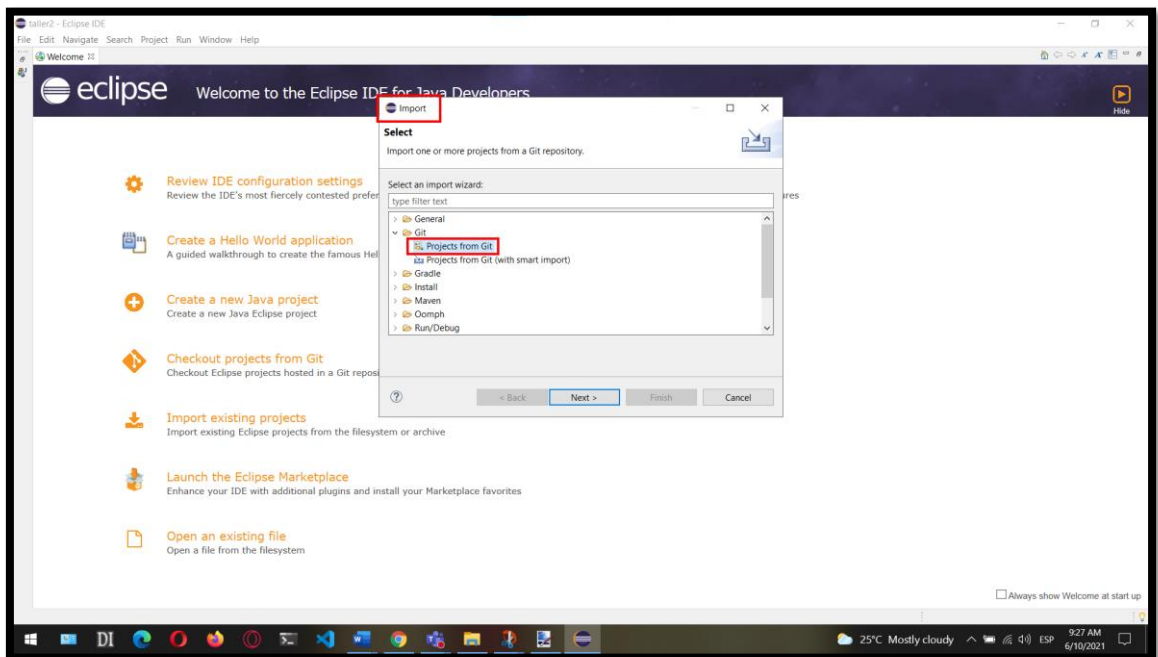
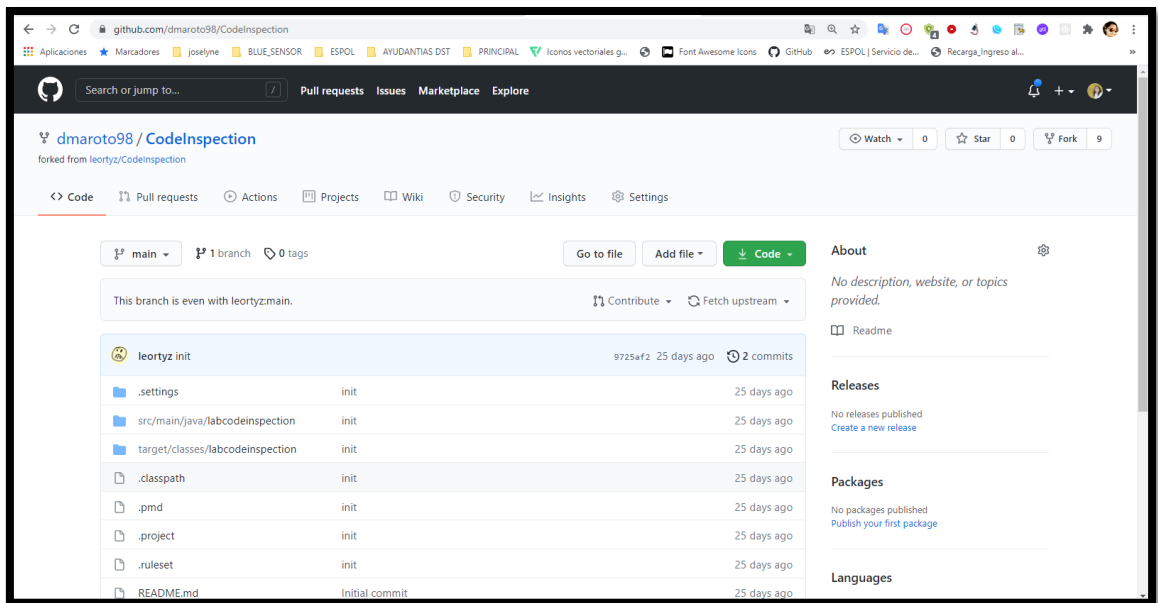


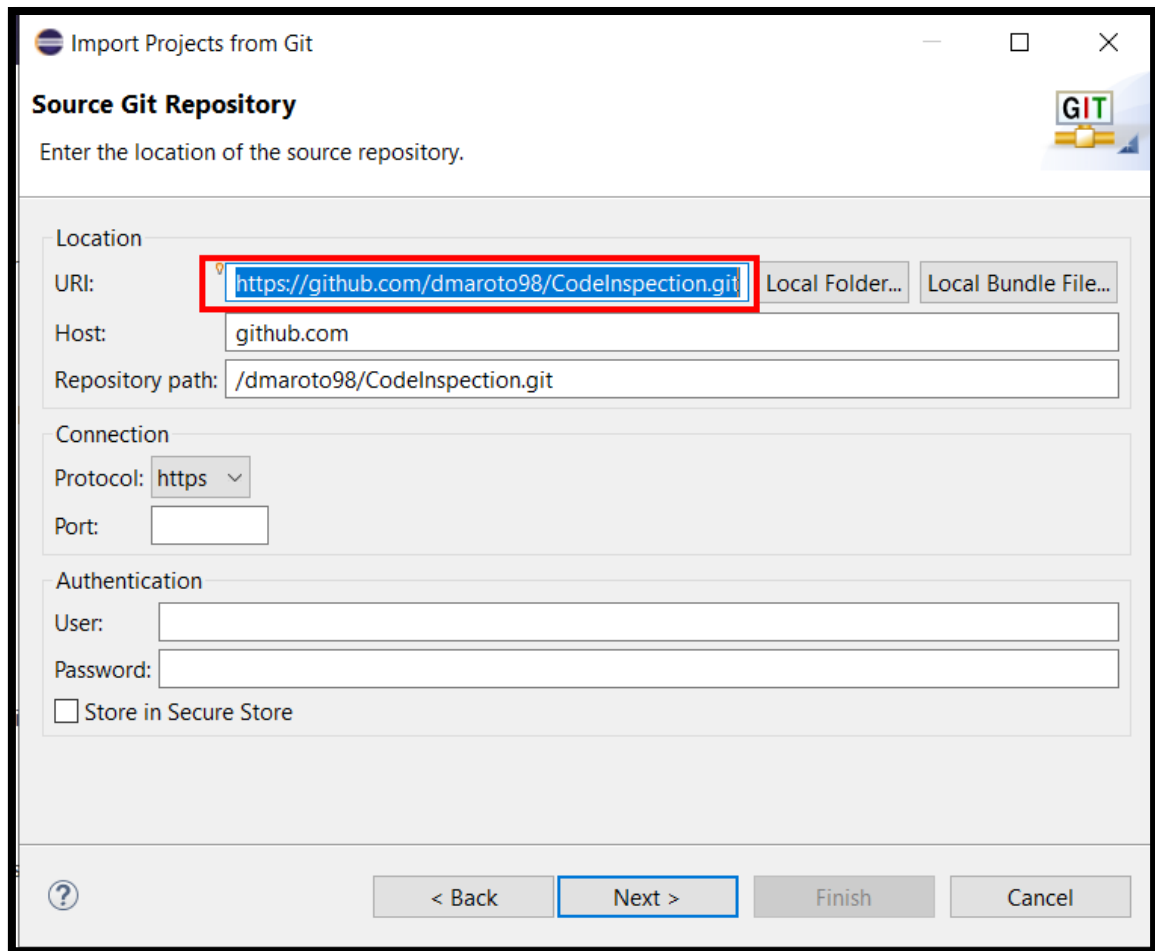


Part 3: Download and configure the project

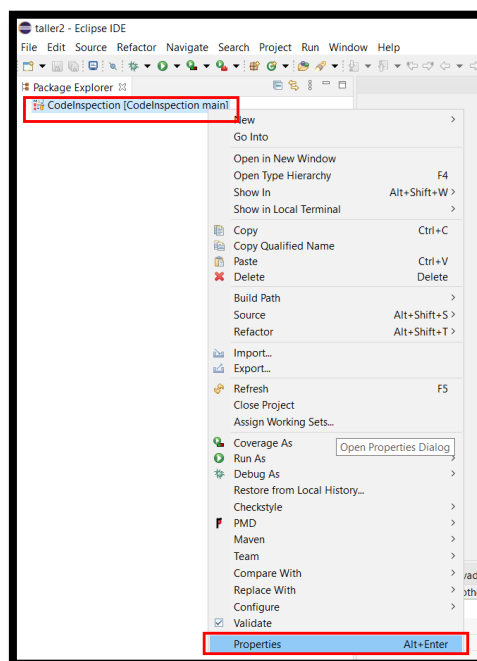
1. Fork the following repository: [CodeInspection](https://github.com/leortyz/CodeInspection) and open the project in eclipse.



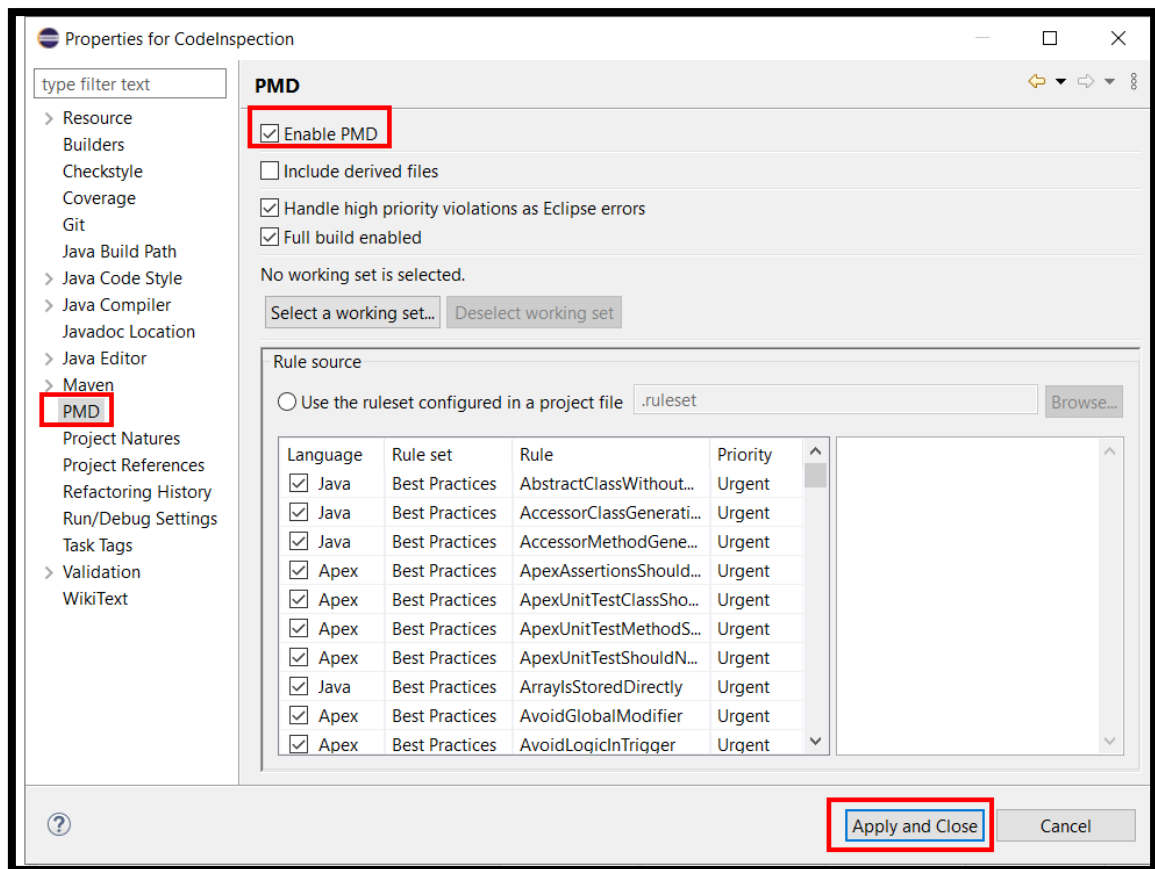




2. PMD will not be activated for the project by default. Open the project properties window by clicking in “Project >> Properties”.

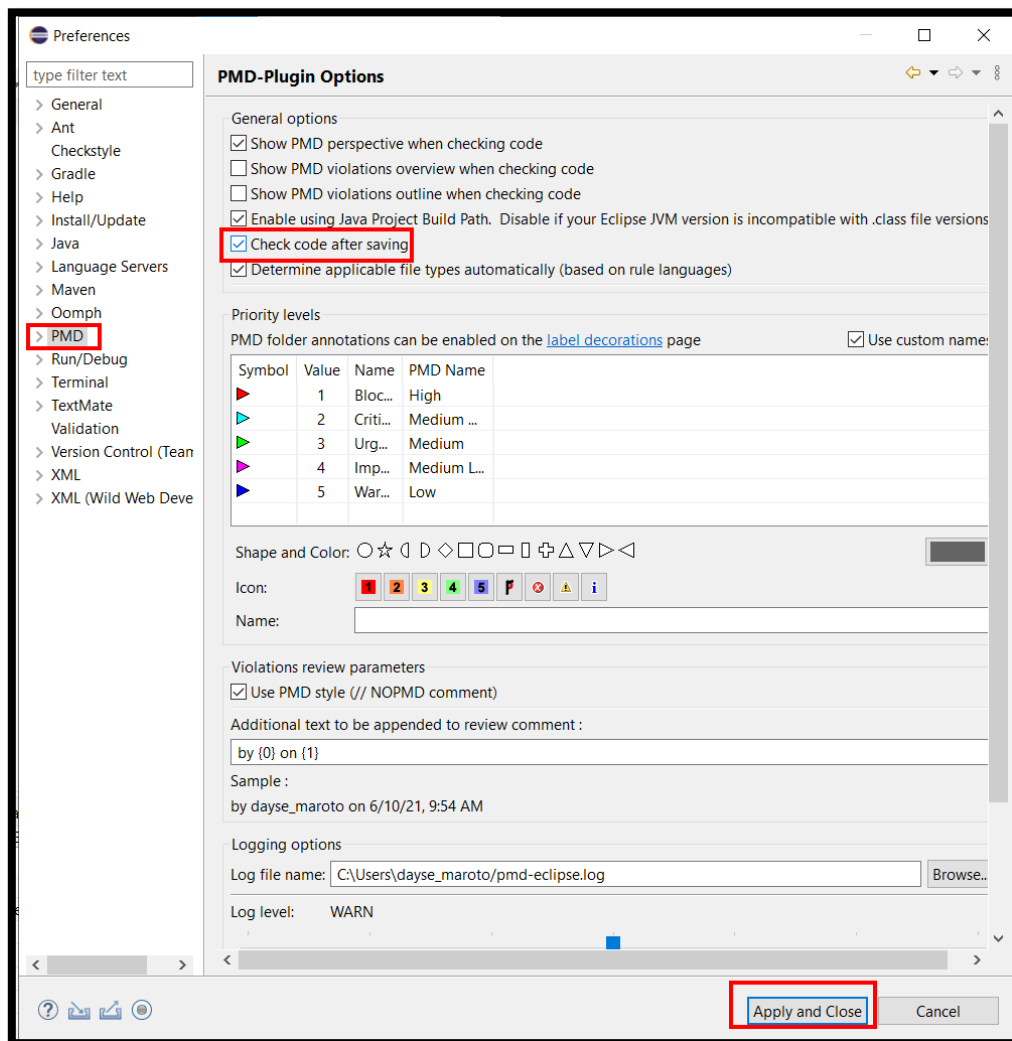


3. Select "PMD" on the side bar and check "Enabled PMD".
4. Look at all the rulesets that come with PMD, leave the default set of rules.
5. Click "Apply and Close" and "Yes".



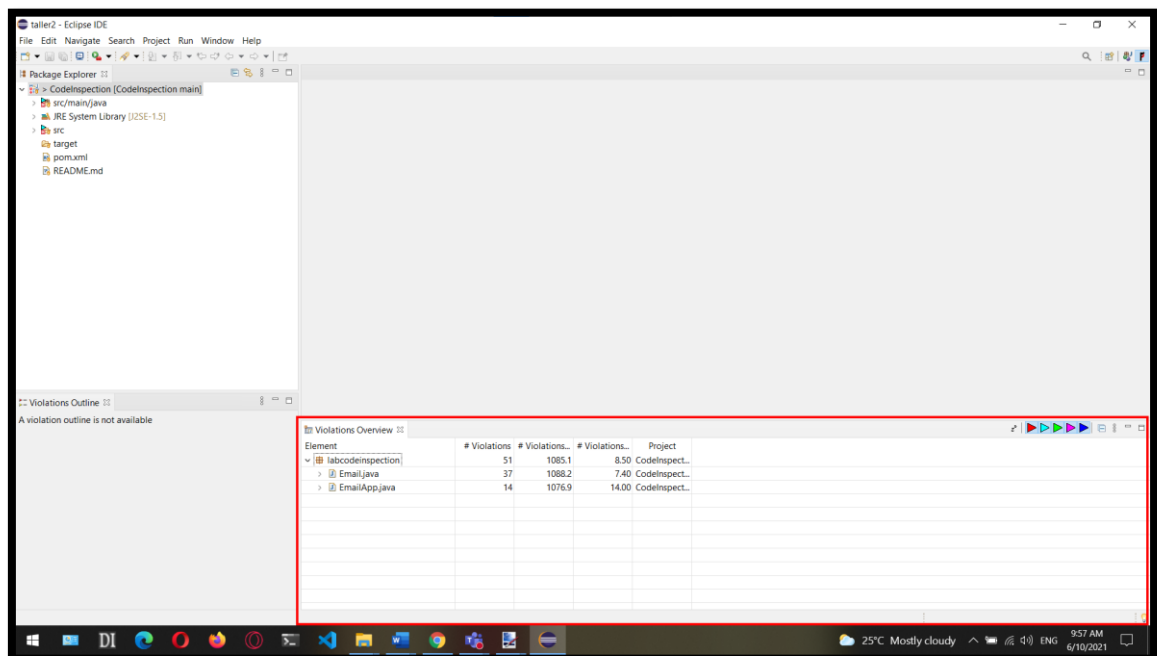
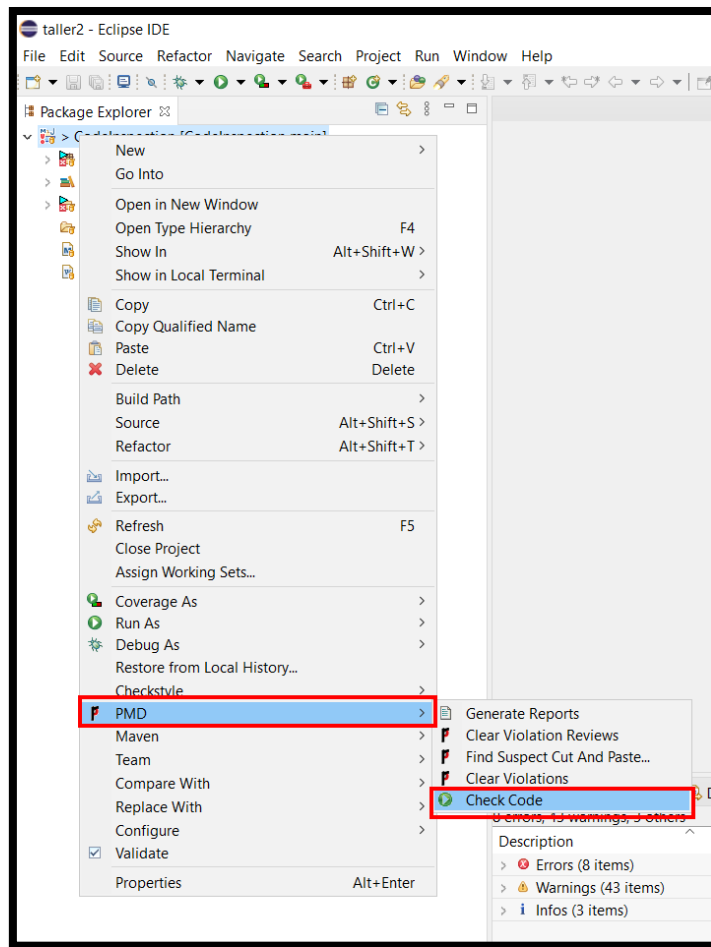
Part 4: Using PMD

Go to “Window >> Preferences”, select “PMD” and check “Check code after saving”



To run PMD, right click on the project and select “PMD >> Check code”
Two new windows are displayed with all violations. Each violation has its priority represented by a color and corresponding rule. The meaning of the colors is:

- Red is blocker >>> High priority.
- Cyan is critical >>> Medium priority.
- Green is urgent >>> Medium priority.
- Pink is Important >>> Medium priority.
- Blue is Warning >>> Low priority.



If you see that files are duplicated in the Violation Overview Window, check code again.

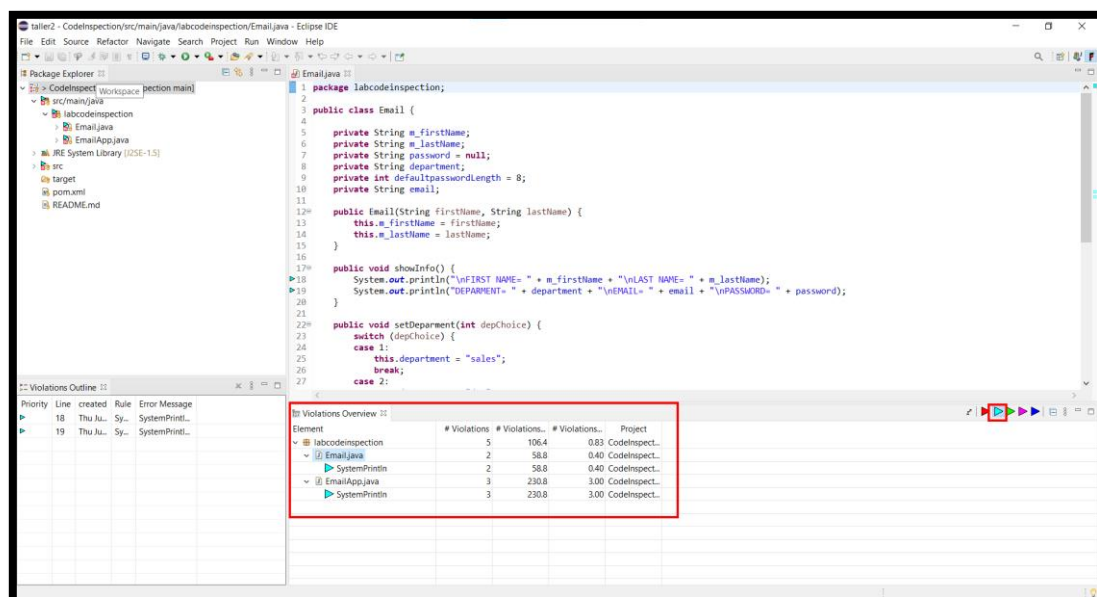
PMD shows the violations next to the lines that generate them.



```
1 package labcodeinspection;
2
3 public class Email {
4
5     private String m_firstName;
6     private String m_lastName;
7     private String password = null;
8     private String department;
9     private int defaultpasswordLength = 8;
10    private String email;
11
12    public Email(String firstName, String lastName) {
13        this.m_firstName = firstName;
14        this.m_lastName = lastName;
15    }
16
17    public void showInfo() {
18        System.out.println("\nFIRST NAME= " + m_firstName + "\nLAST NAME= " + m_lastName);
19        System.out.println("DEPARTMENT= " + department + "\nEMAIL= " + email + "\nPASSWORD= " + password);
20    }
21
22    public void setDepartment(int depChoice) {
23        switch (depChoice) {
24            case 1:
25                this.department = "sales";
26                break;
27            case 2:
```

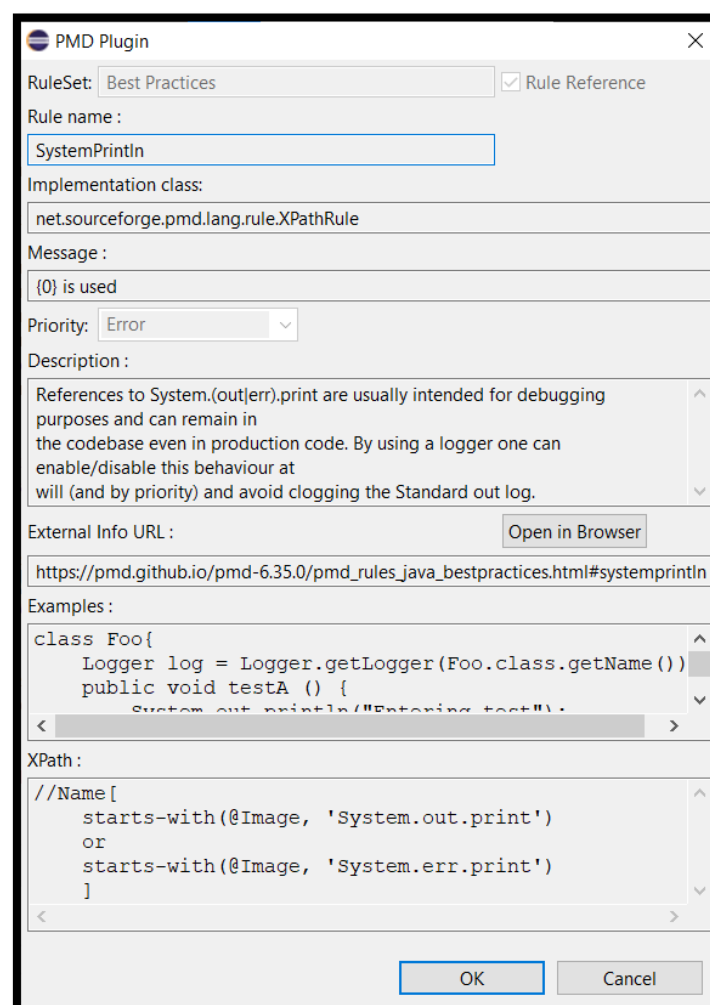
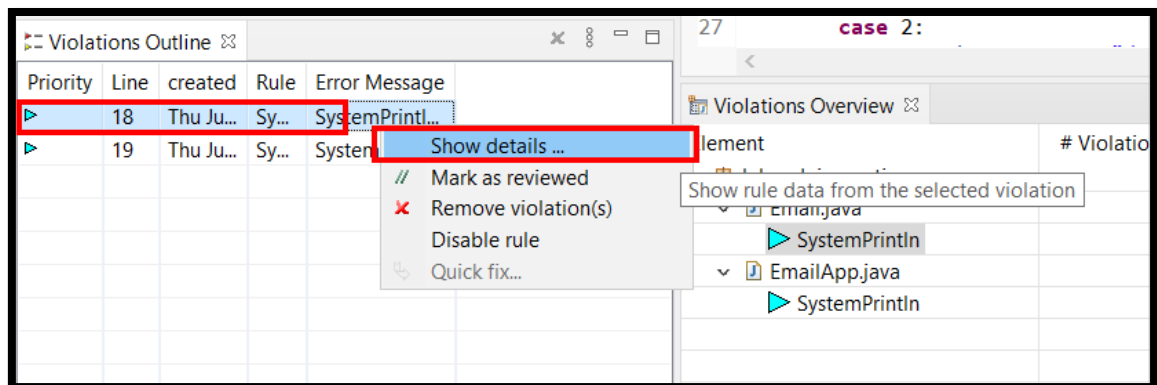
We can filter violations using the color indicators on the top right of Violations Overview window.

For example, if we filter only by critical violations, Violations Overview window shows that Email.java and EmalApp.java have 2 and 3 violations respectively for the rule “SystemPrintln”. If we double-click on an element, the Violation Outline window will update showing all errors related to a file and some important data such as the violation line, the affected rule, and the error message.



Element	# Violations	# Violations...	# Violations...	Project
labcodeinspection	5	106.4	0.83	CodeInspect...
Email.java	2	58.8	0.40	CodeInspect...
SystemPrintln	2	58.8	0.40	CodeInspect...
EmailApp.java	3	230.8	3.00	CodeInspect...
SystemPrintln	3	230.8	3.00	CodeInspect...

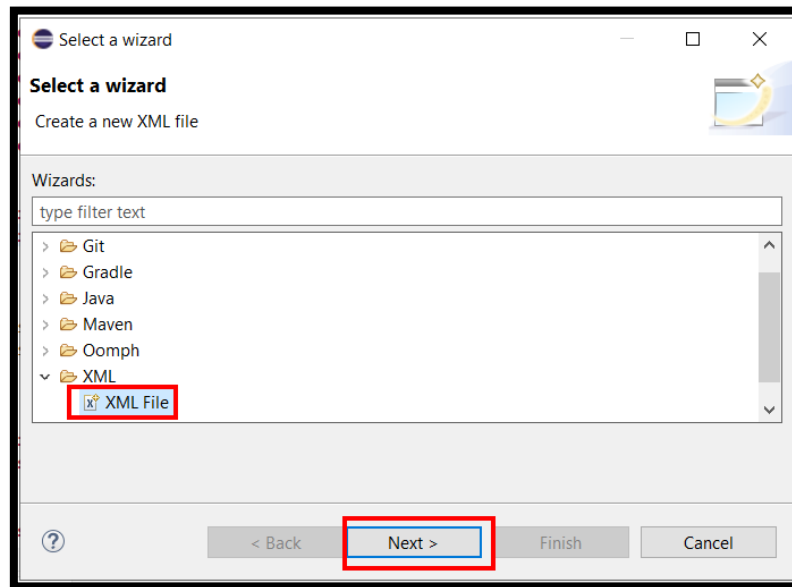
We can right-click on a violation and select "Show details..." for more information and an example solution.



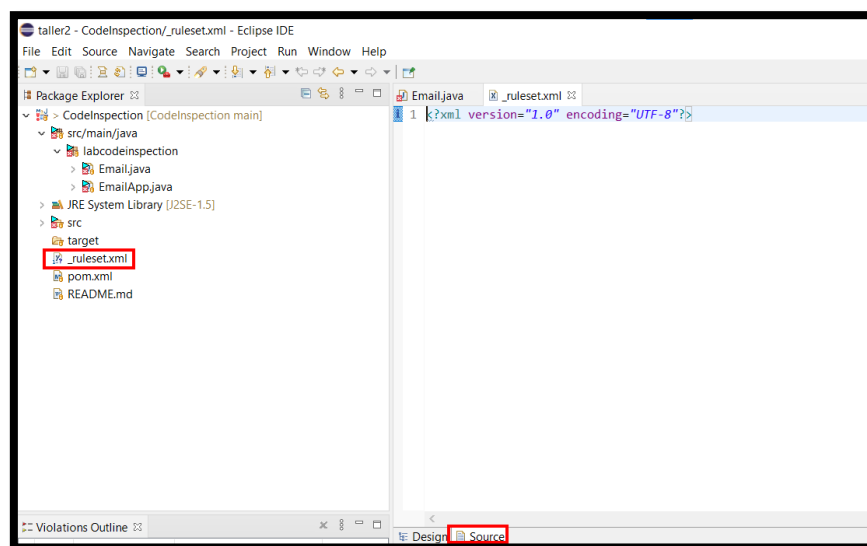
Part 5: Creating PMD rulesets

A PMD ruleset is simply an XML file that lists a set of rules that fit the project. You can include entire rulesets, or selectively choose specific rules from within other rulesets. You can also provide extra parameters to certain rules to customize their behavior. To do so, follow these steps:

1. Right click on the project, then “New >> Other >> XML >> XML File”.



2. Select the project, enter a file name as “_ruleset “
3. Click “Source” in the bottom tab to change the view and edit the file directly.



4. Here is a fragment of a typical configuration document, copy and paste into the file:

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleset name="Dayse Maroto Rules"
  xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0
https://pmd.sourceforge.io/ruleset_2_0_0.xsd">

    <description>
        Code Inspection Lab, Dayse Joselyne Maroto Lema
    </description>

</ruleset>
```

5. Let's reference a complete ruleset. Add the following line below the description tag:

```
<rule ref="category/java/performance.xml" />
```

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleset name="Dayse Maroto Rules"
  xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0
https://pmd.sourceforge.io/ruleset_2_0_0.xsd">
    <description>
        Code Inspection Lab, Dayse Joselyne Maroto Lema
    </description>
    <rule ref="category/java/performance.xml" />

</ruleset>
```

6. Now, add another reference, but exclude some rules from the ruleset:

```
<rule ref="category/java/bestpractices.xml">
    <exclude name="SystemPrintln" />
</rule>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleset name="Dayse Maroto Rules"
  xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0
https://pmd.sourceforge.io/ruleset_2_0_0.xsd">

    <description>
```

```

        Code Inspection Lab, Dayse Joselyne Maroto Lema
    </description>
    <rule ref="category/java/performance.xml" />
    <rule ref="category/java/bestpractices.xml">
        <exclude name="SystemPrintln" />
    </rule>
</ruleset>

```

This ruleset also comes by default with PMD and it has rules which enforce generally accepted best practices but excluding “SystemPrintln” rule.

7. We can add rules from a specific ruleset as follow:

```

<rule ref="category/java/design.xml/ImmutableField" />
<rule ref="category/java/design.xml/UseUtilityClass">
    <priority>1</priority>
</rule>

```

```

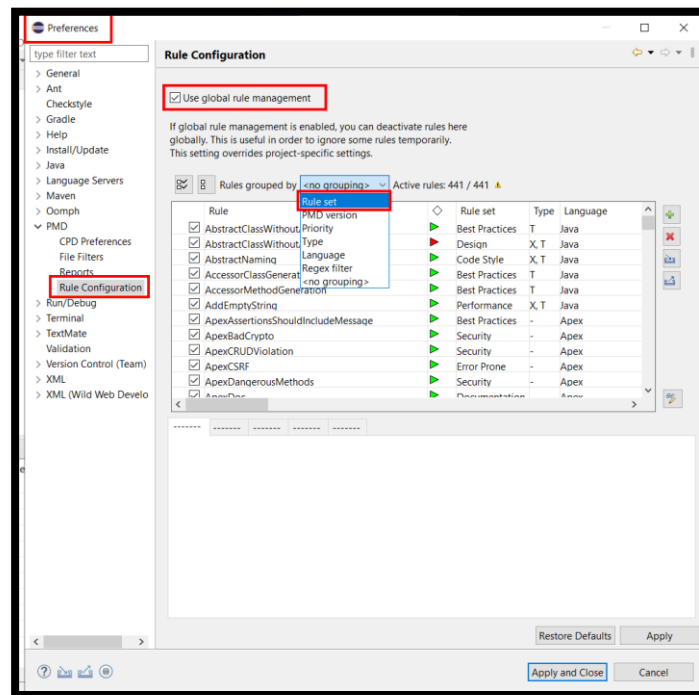
<?xml version="1.0" encoding="UTF-8"?>
<ruleset name="Dayse Rules"
    xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0
https://pmd.sourceforge.io/ruleset_2_0_0.xsd">
    <description>
        Code Inspection Lab, Dayse Joselyne Maroto Lema
    </description>
    <rule ref="category/java/performance.xml" />
    <rule ref="category/java/bestpractices.xml">
        <exclude name="SystemPrintln" />
    </rule>
    <rule ref="category/java/design.xml/ImmutableField" />
    <rule ref="category/java/design.xml/UseUtilityClass">
        <priority>1</priority>
    </rule>
</ruleset>

```

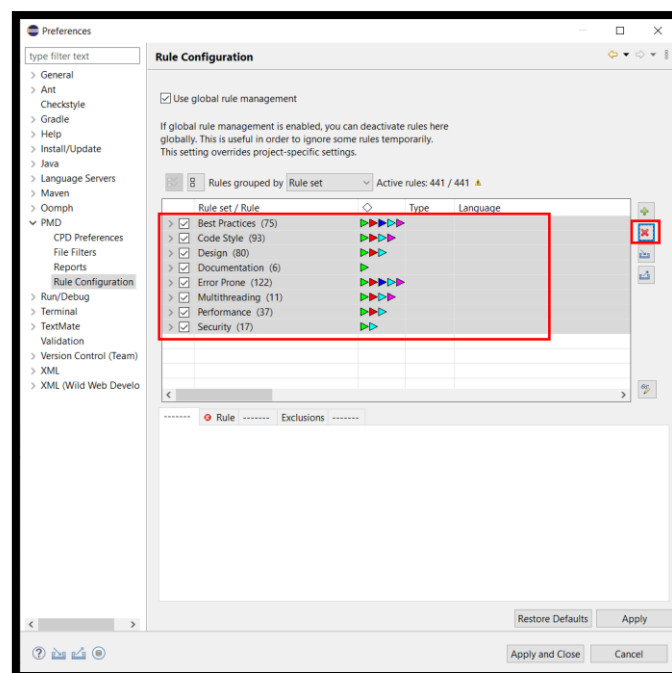
Here, we are adding “ImmutableField” rule and “UseUtilityClass” rule from the Design ruleset and changing its priority to 1. Priority is an integer ranging from 1 to 5, with 1 being the highest priority.

For more information about PMD rulesets, refer to PMD documentation [3]

8. To use an external ruleset, we need to go to the PMD Configuration Window. Go to “Window >> Preferences ◇ PMD ◇ Rule Configurations”.
9. Check “Use global rule management”, then group rules by “Rule Set”.

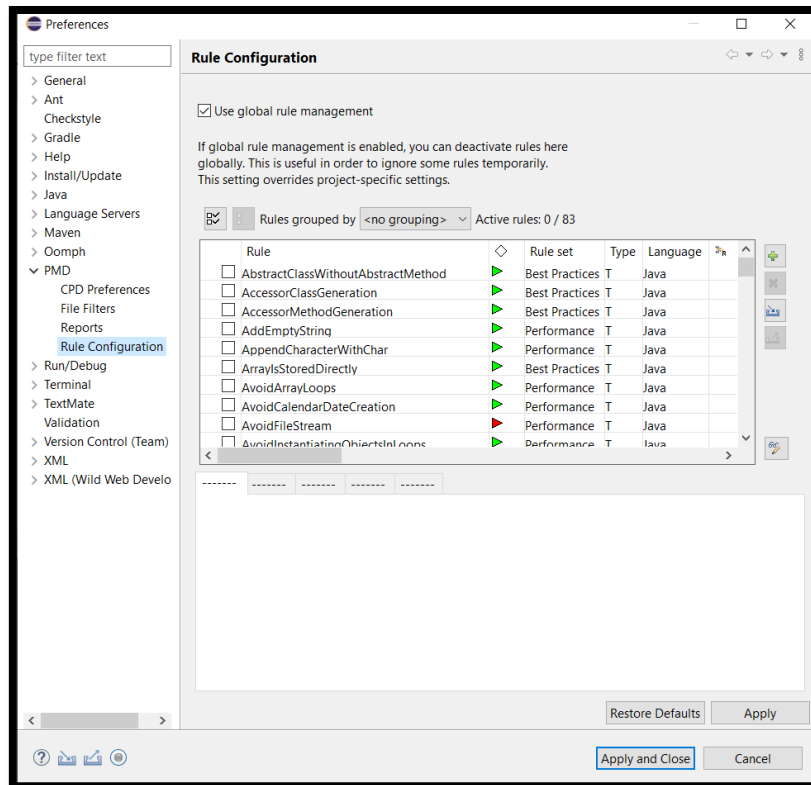


10. Select all the rule sets with shift and select one by one and click in the “X” button to delete them.

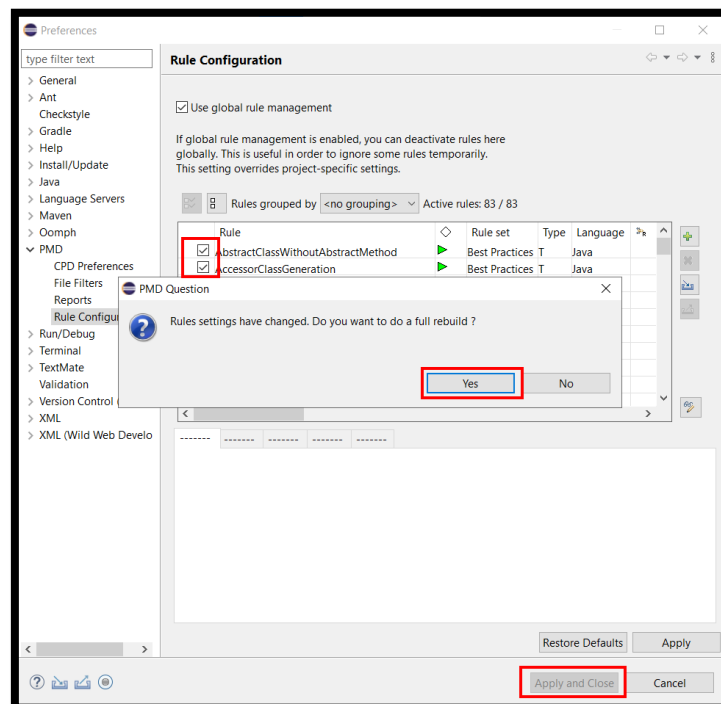


11. Click "Import rule set..." (under the "x"), browse your file and Click “Ok”.

13. Return to PMD Configuration Window. Notice that the checkbox next to the rule names is unchecked.

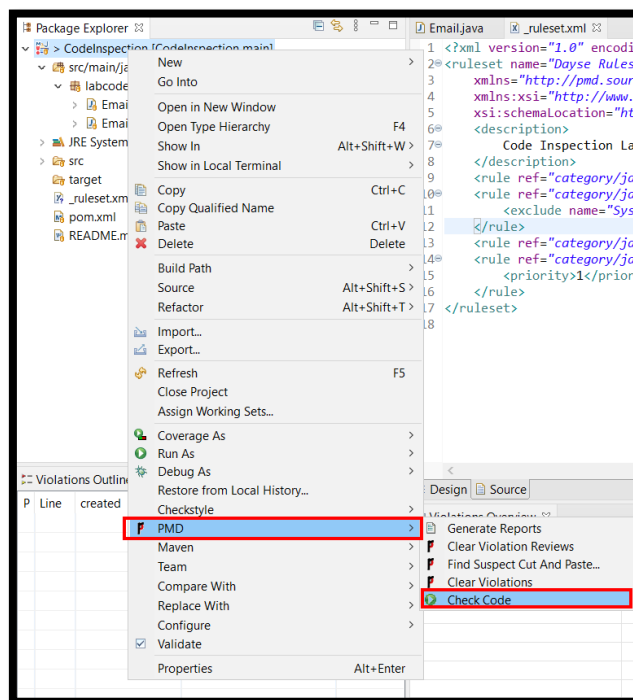
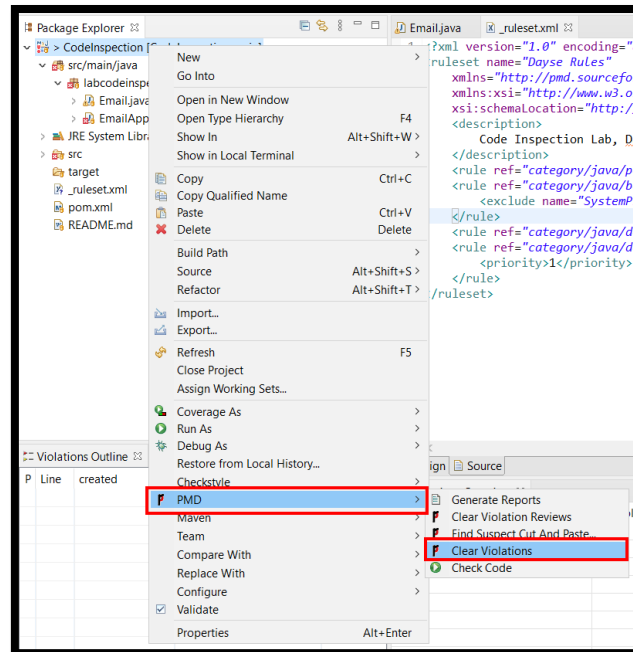


14. Press "Apply and close", then "Yes".



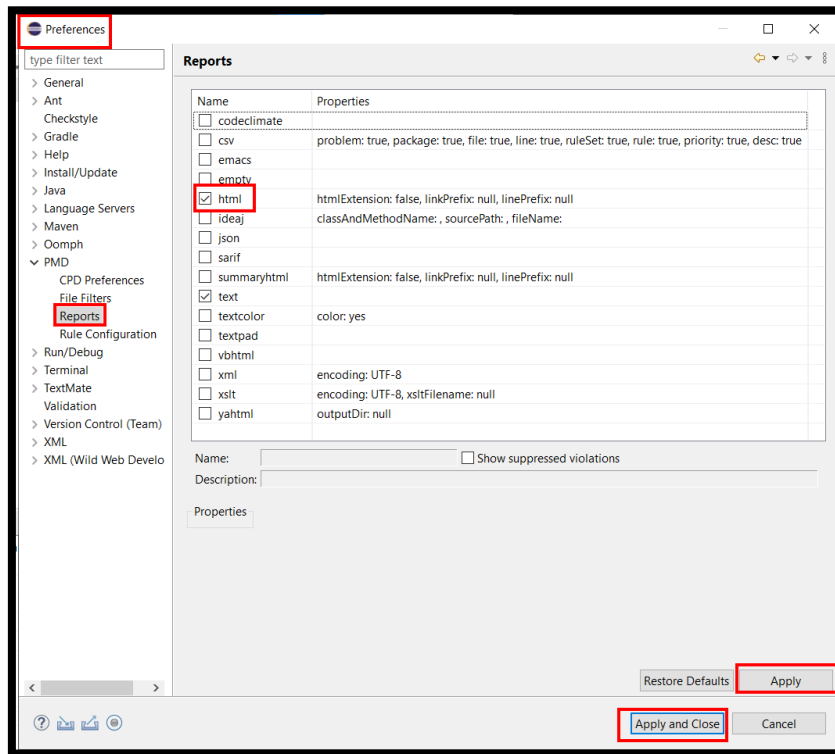
15. Right click on the project and select “PMD >> Clear Violations”. Then “PMD > Check Code.”

Note that it is possible to configure the properties for each rule we add. To learn more about rulesets and their properties, refer to PMD Java Rules [3]

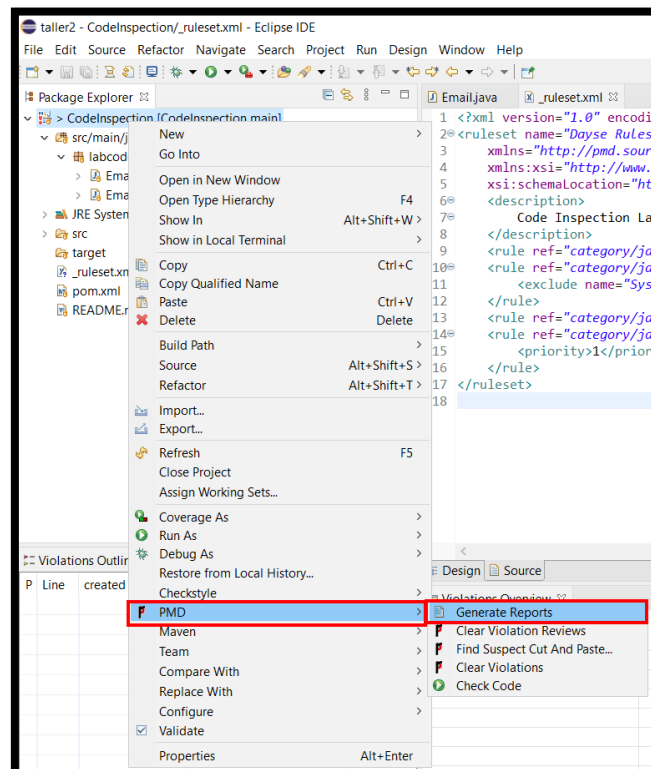


Part 6: Generating a PMD report

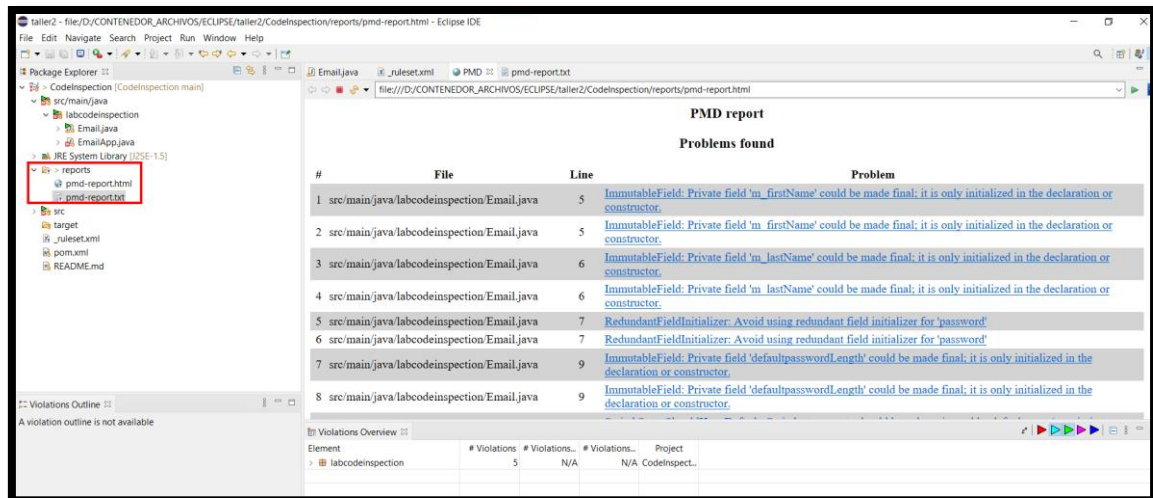
1. Open PMD configuration window, select Reports and check “html.”



2. Right-click the project then click “PMD ◇ Generate Report.”



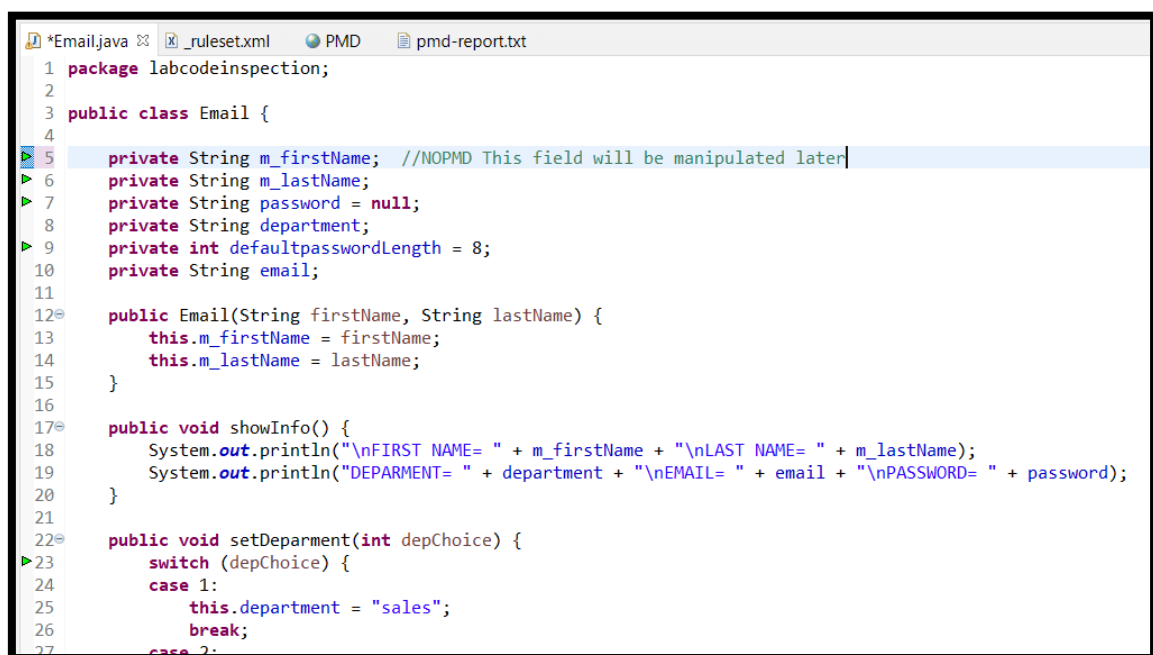
3. A folder named reports is created in the tree project. Open it and double click the “pmd-report.html” to see the full report.



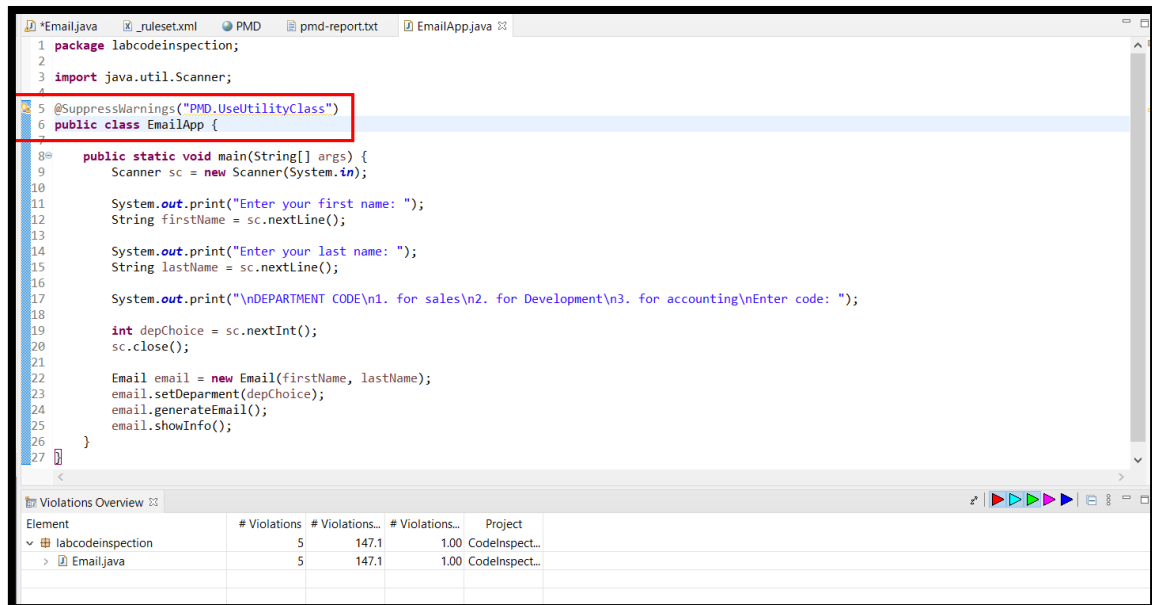
Part 7: Suppressing PMD Rules

Sometimes you will have a legitimate reason for not respecting one of the PMD rules. PMD provides several methods by which Rule violations can be suppressed. We will be using comments and annotations.

1. Go to Email.java. See that line 5 has a violation related to ImmutableField rule.
2. Write “NOPMD” as a comment in the same line where the violation occurred.
3. Optionally, add a message placed after the NOPMD marker. This will get placed in the report.



- Go to EmailApp.java and check the violation. The rule violated is “UseUtilityClass”.
- Write an annotation above the line as follow:

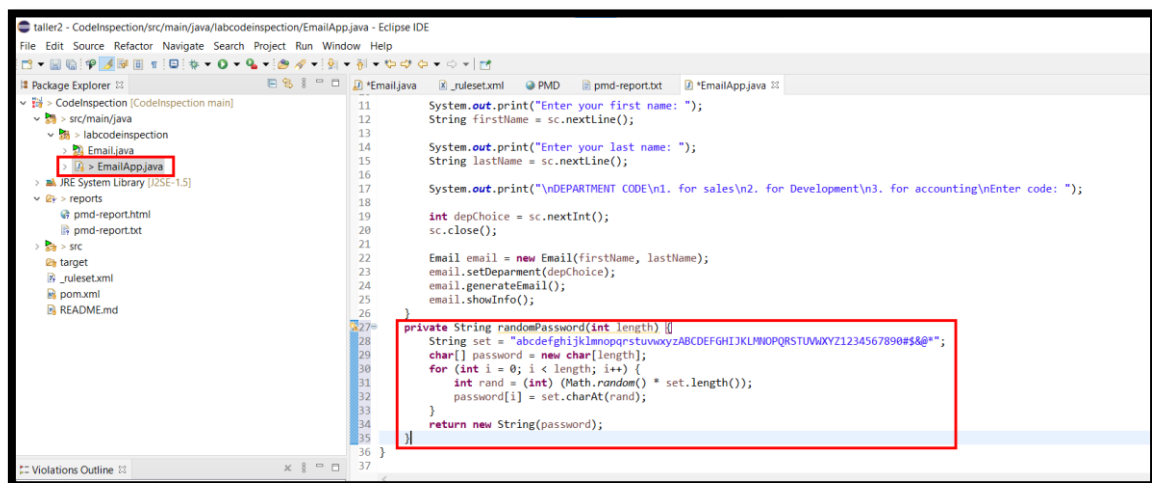


Please note that only that rule will be ignored. 6. Save the file and see the results.

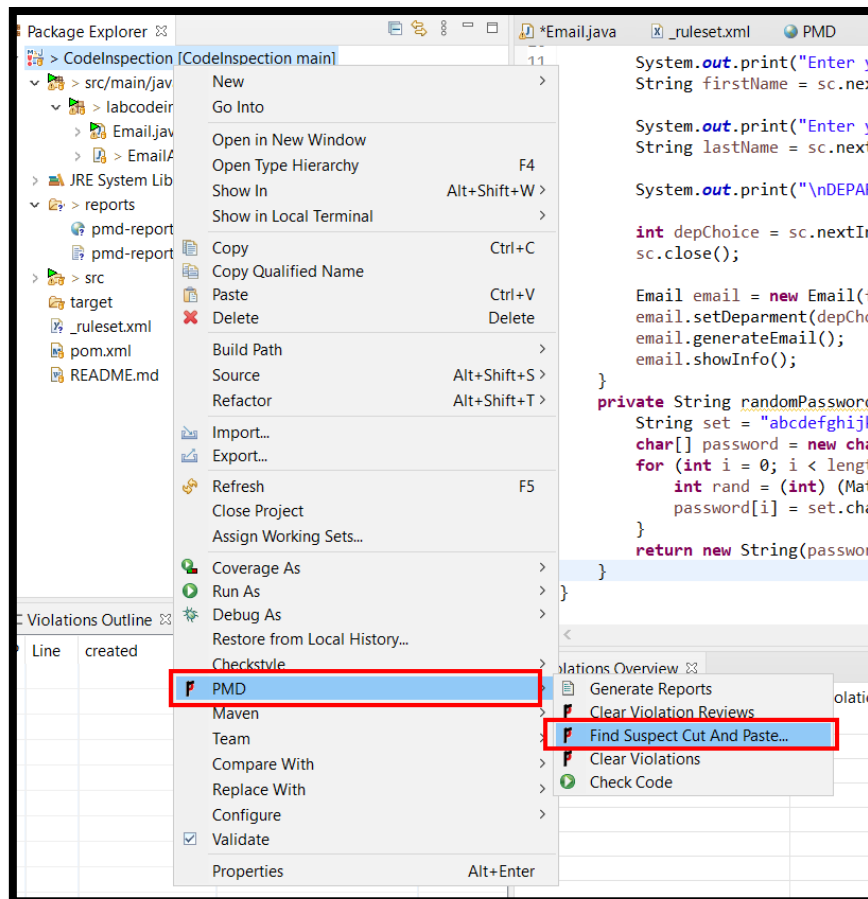
Part 8: Detecting Cut-and-Paste with CPD

PMD comes with a useful tool for detecting cut-and-pasted code called CPD (Cut-and-Paste Detector). Follow these steps to use it and generate a report.

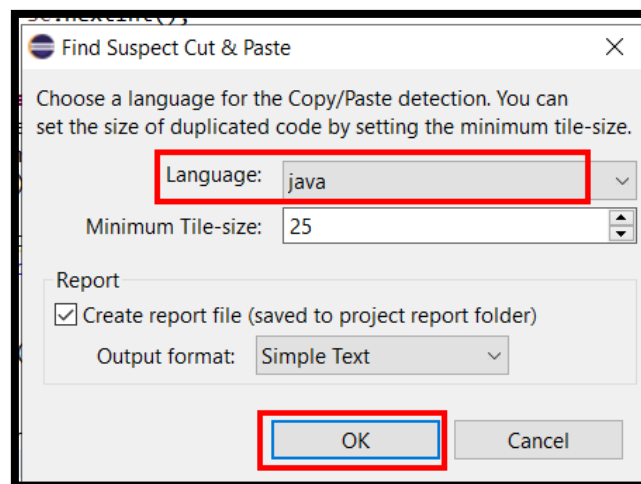
- Just for demonstration purpose, copy and paste the “randomPassword” method from Email.java to EmailApp.java.



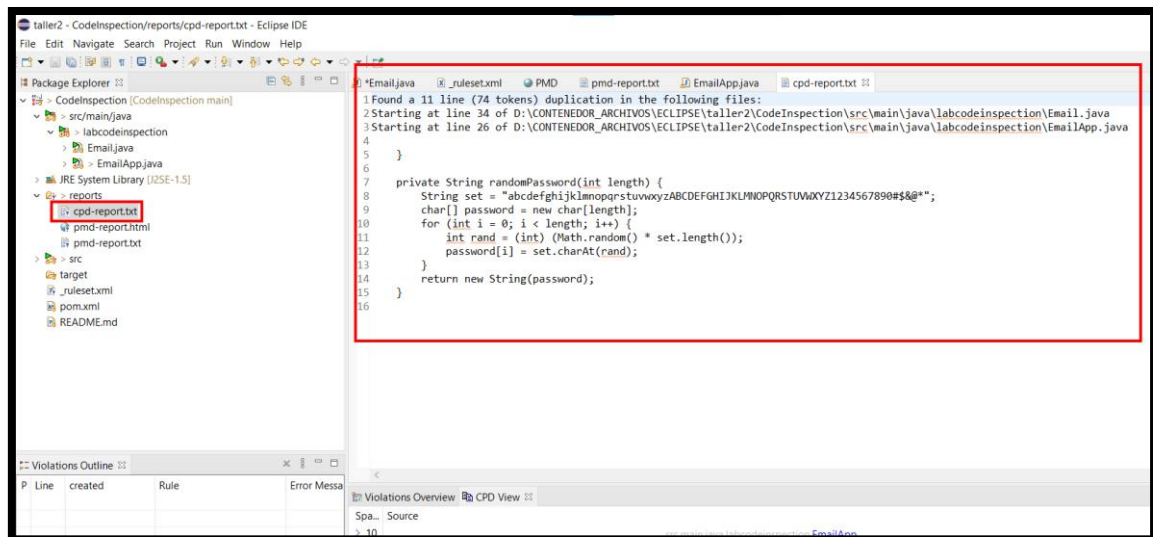
- Right-click the project and select “PMD >> Find Suspect Cut and Paste” from menu options.



3. Select “java” for Language, then click “Ok”.



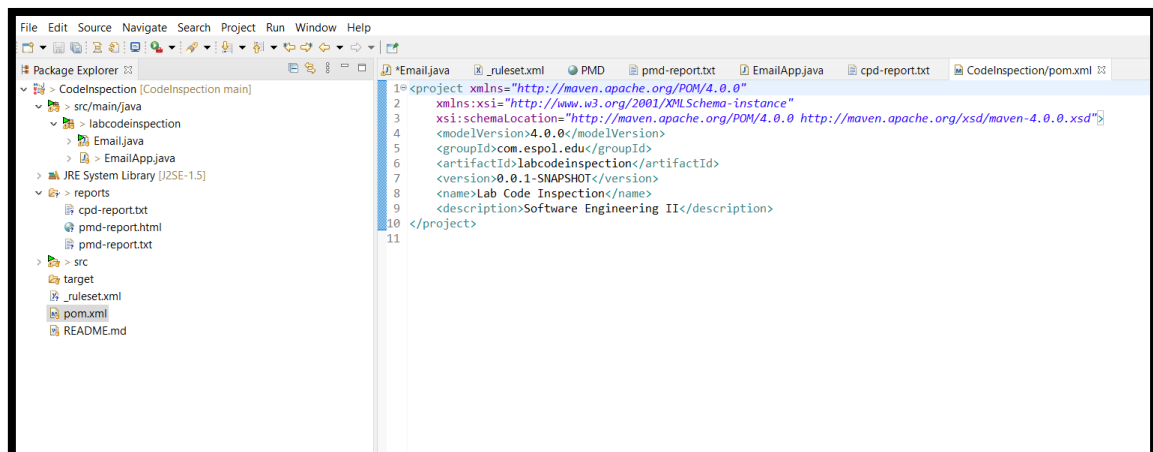
4. CPD View Window will open with the results. Also, a text file called cpd-report.txt will be generated in the /report directory.



5. Revert the changes made in EmailApp.java.

Part 9: PMD and Maven

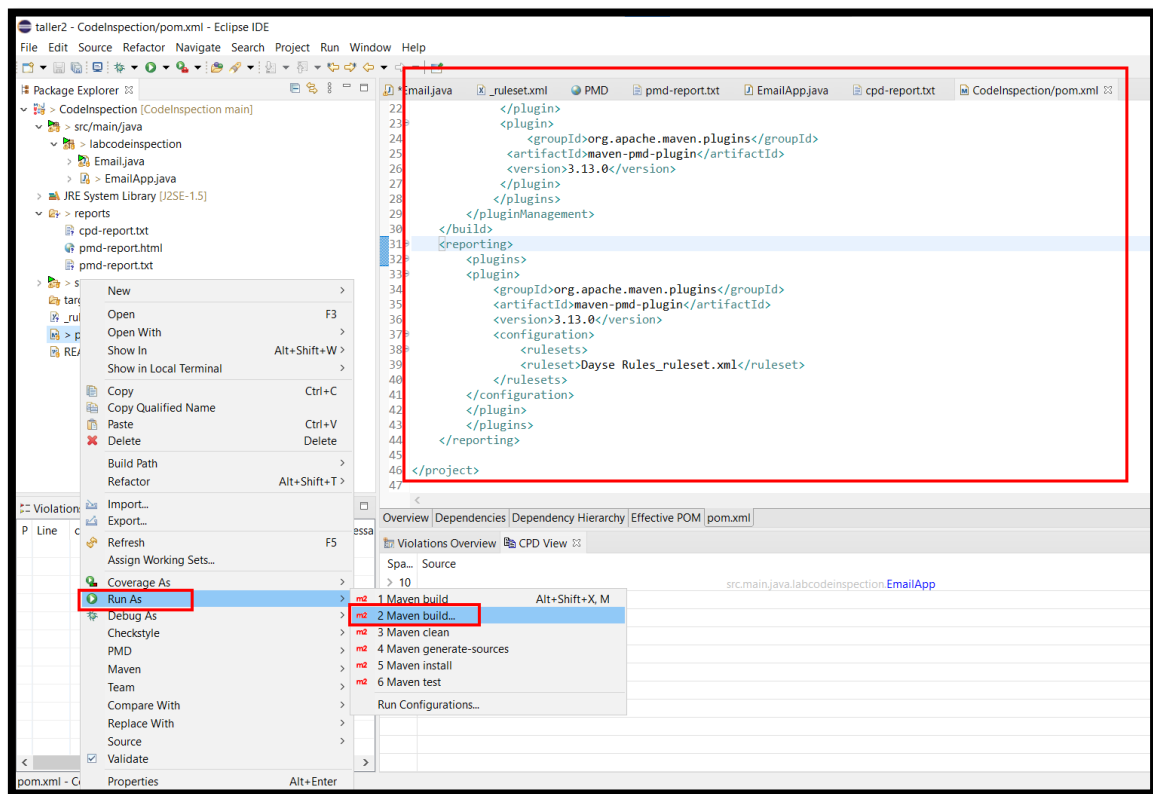
1. Open “pom.xml” file



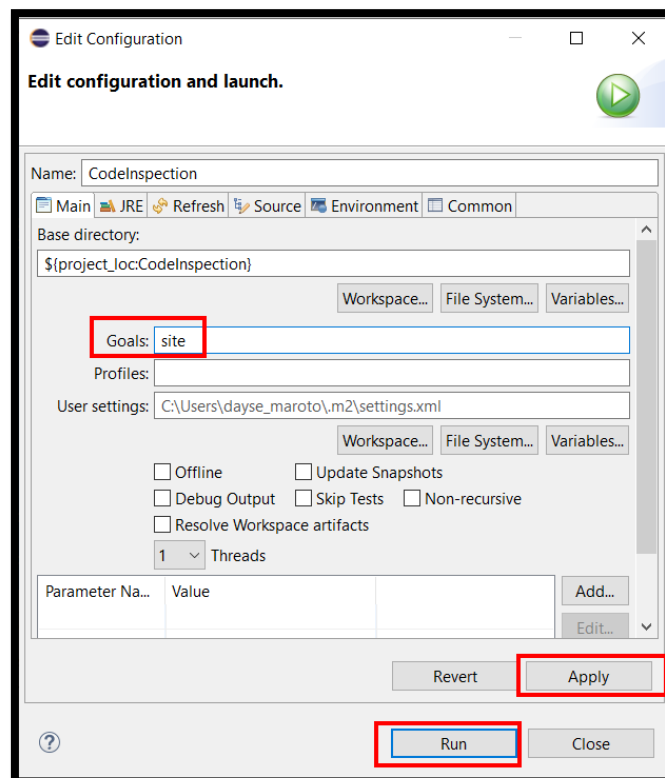
2. Add the following lines under project tag to install all the necessary plugins:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.espol.edu</groupId>
  <artifactId>labcodeinspection</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Lab Code Inspection</name>
  <description>Software Engineering II</description>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-site-plugin</artifactId>
          <version>3.7.1</version>
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-project-info-reports-
plugin</artifactId>
          <version>3.0.0</version>
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-pmd-plugin</artifactId>
          <version>3.13.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
        <version>3.13.0</version>
        <configuration>
          <rulesets>
            <ruleset>Dayse Rules_ruleset.xml</ruleset>
          </rulesets>
        </configuration>
      </plugin>
    </plugins>
  </reporting>
</project>
```

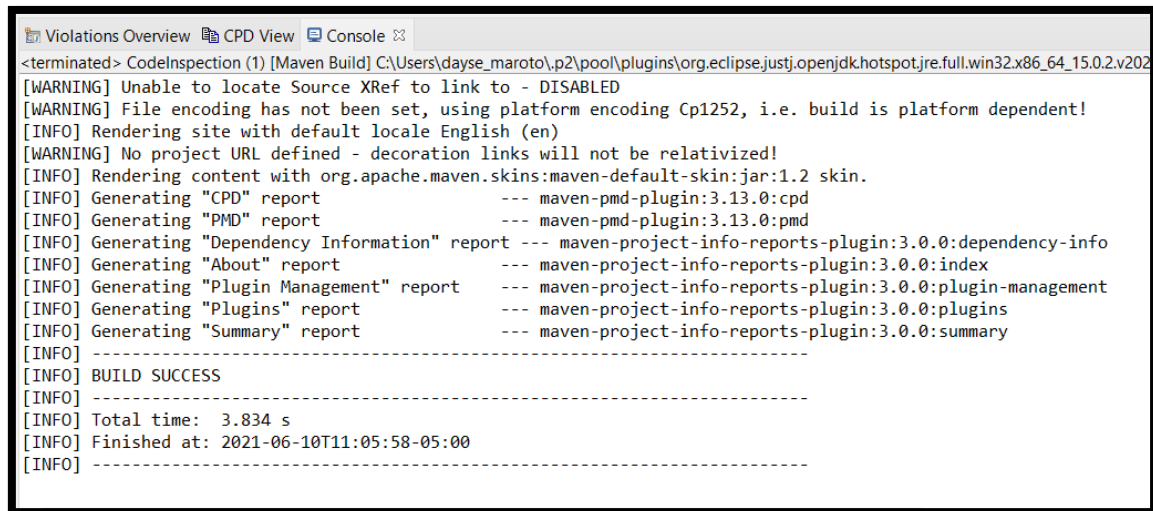

The ruleset tag is used to specify a file that contains rules to use in the checking process. In this case, we are telling the plugin to use our ruleset file. 3. Save the file, right Click on it, then “Run as >> Maven build...”



4. Type “site” for the Goals and Click Run.



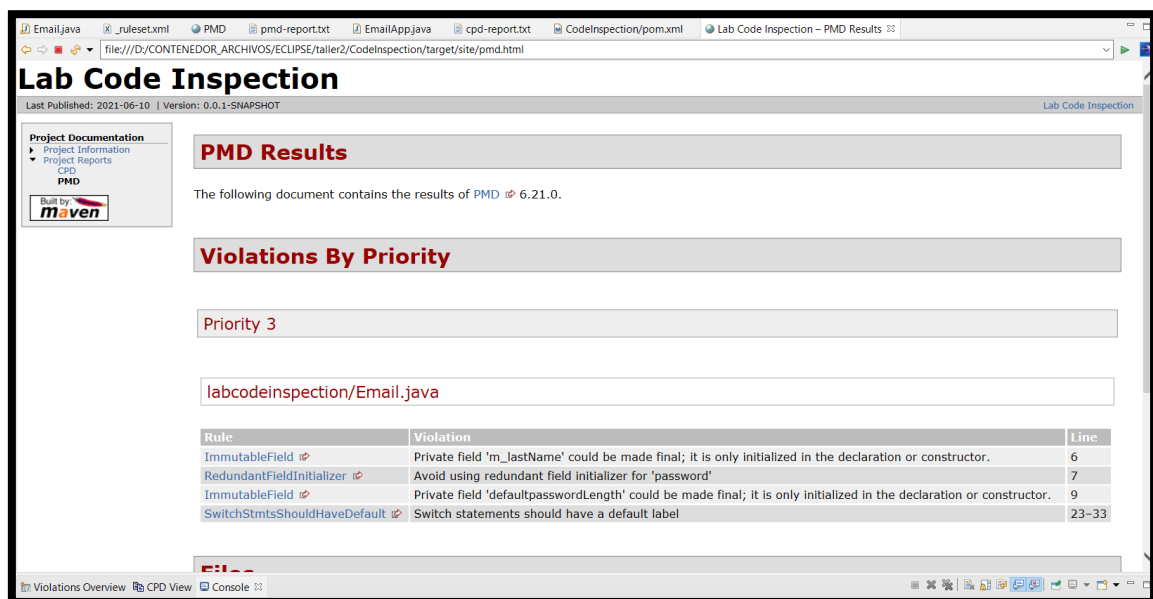
- Wait for the process to finish.



The screenshot shows the Eclipse IDE Console window with the following output:

```
<terminated> CodeInspection (1) [Maven Build] C:\Users\dayse_maroto\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210610
[WARNING] Unable to locate Source XRef to link to - DISABLED
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Rendering site with default locale English (en)
[WARNING] No project URL defined - decoration links will not be relativized!
[INFO] Rendering content with org.apache.maven.skins:maven-default-skin:jar:1.2 skin.
[INFO] Generating "CPD" report --- maven-pmd-plugin:3.13.0:cpd
[INFO] Generating "PMD" report --- maven-pmd-plugin:3.13.0:pmd
[INFO] Generating "Dependency Information" report --- maven-project-info-reports-plugin:3.0.0:dependency-info
[INFO] Generating "About" report --- maven-project-info-reports-plugin:3.0.0:index
[INFO] Generating "Plugin Management" report --- maven-project-info-reports-plugin:3.0.0:plugin-management
[INFO] Generating "Plugins" report --- maven-project-info-reports-plugin:3.0.0:plugins
[INFO] Generating "Summary" report --- maven-project-info-reports-plugin:3.0.0:summary
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.834 s
[INFO] Finished at: 2021-06-10T11:05:58-05:00
[INFO] -----
```

- Go to your project directory, open "target>> site". This folder is visible from the tree project in Eclipse too.
- Several html files are shown. Open "index.html".



The screenshot shows a web browser displaying the "Lab Code Inspection" results page. The page title is "Lab Code Inspection" and it includes a sidebar with "Project Documentation" and "PMD" links. The main content area shows "PMD Results" and "Violations By Priority".

PMD Results

The following document contains the results of PMD 6.21.0.

Violations By Priority

Priority 3

labcodeinspection/Email.java

Rule	Violation	Line
ImmutableField	Private field 'm_LastName' could be made final; it is only initialized in the declaration or constructor.	6
RedundantFieldInitializer	Avoid using redundant field initializer for 'password'	7
ImmutableField	Private field 'defaultpasswordLength' could be made final; it is only initialized in the declaration or constructor.	9
SwitchStmtsShouldHaveDefault	Switch statements should have a default label	23-33

2) Development

1. Delete all comments and annotations from the code.
2. Add the following ruleset and rules:
 - Code Style Ruleset
 - Rule “BeanMembersShouldSerialize” from Error Prone ruleset with a priority of 2
 - Rule “UseLocaleWithCaseConversion” from Error Prone ruleset
 - Rule “CommentRequired” from Documentation ruleset with these properties set to “Ignored”:
 - classCommentRequirement
 - headerCommentRequirement
 - fieldCommentRequirement

TIP: Refer to the PMD Java Rules [3].

The errors per class are show below:

Element	# Violations
labcodeinspection	35
Email.java	26
EmailApp.java	9

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset name="Dayse Rules"
  xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0
  https://pmd.sourceforge.io/ruleset_2_0_0.xsd">
  <description>
    Code Inspection Lab, Dayse Joselyne Maroto Lema
  </description>
  <rule ref="category/java/performance.xml" />
  <rule ref="category/java/bestpractices.xml">
    <exclude name="SystemPrintLn" />
  </rule>
  <rule ref="category/java/design.xml/ImmutableField" />
  <rule ref="category/java/design.xml/UseUtilityClass">
    <priority>1</priority>
  </rule>

  <rule ref="category/java/codestyle.xml" />

  <rule
ref="category/java/errorprone.xml/BeanMembersShouldSerialize">
    <priority>2</priority>
  </rule>
  <rule
ref="category/java/errorprone.xml/UseLocaleWithCaseConversions" />
```

```

<rule ref="category/java/documentation.xml/CommentRequired">
  <properties>
    <property name="classCommentRequirement"
value="Ignored" />
    <property name="headerCommentRequirement"
value="Ignored" />
    <property name="fieldCommentRequirement"
value="Ignored" />
  </properties>
</rule>

</ruleset>

```

The screenshot shows the Eclipse IDE with the PMD configuration file `_ruleset.xml` open. The configuration includes rules for performance, best practices, design, and code style. The `CommentRequired` rule is configured to ignore class, header, and field comments.

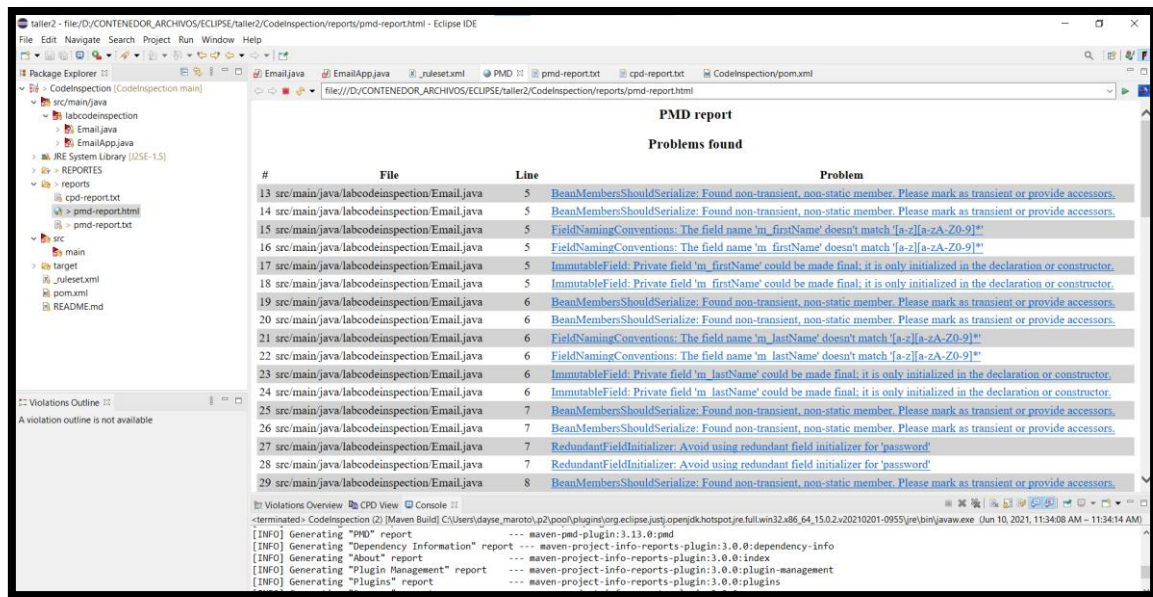
The **Violations Outline** tab shows a table of violations:

Element	# Violations	# Violations/KLOC	# Violations/Method	Project
labcodeinspection	35	744.7	5.83	CodeInspection
EmailApp.java	9	692.3	9.00	CodeInspection
Email.java	26	764.7	5.20	CodeInspection

3. Generate an html report, make a copy and save it somewhere on your disk.

The screenshot shows the PMD HTML report generated for the project. The report is titled "Lab Code Inspection" and "PMD Results". It lists the violations by priority, showing two violations of priority 1 in the `labcodeinspection/Email.java` file.

Rule	Violations	Line
FieldNamingConventions	The field name "m_firstName" doesn't match "[a-zA-Z0-9]"	5
FieldNamingConventions	The field name "m_lastName" doesn't match "[a-zA-Z0-9]"	6



4. Correct any violation in the code that have been generated in the report.
5. Generate a new report (without violations).

