

Objectives:

- Use PMD as a code quality metric tool for preemptive defect detection.
- Obtain an html report with the different detected issues in the code using Maven.

Requirements

- Java 8 or newer JRE/JDK
- Eclipse IDE for Java Developers [Download from Eclipse](#) || [Get executable](#)
- Git

Introduction

PMD is a **source code analyzer**. It finds **common programming flaws** like unused variables, empty catch blocks, unnecessary object creation, and so forth [1]. Like other tools, PMD can **verify that coding conventions and standards are followed**. PMD is more focused on **preventive defect detection**. It comes with a vast set of rules and is highly configurable. PMD can also configure - in a simple way - particular rules to use in a specific project [2].

PMD integrates well with IDEs such as Eclipse and NetBeans, and it also fits well into the build process thanks to its smooth integration with Ant and Maven [2].

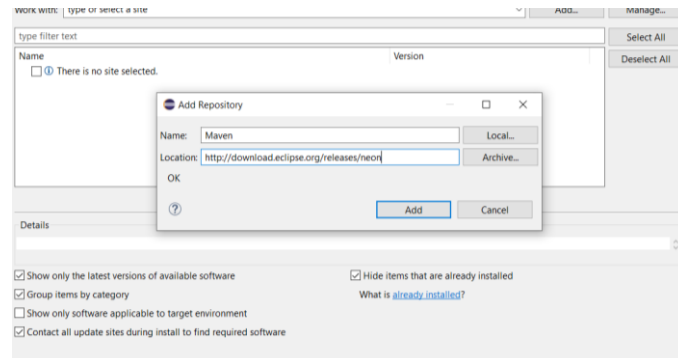
For this lab, we will use **Eclipse and Maven** together with **PMD** to inspect the source code of a project.

Activity

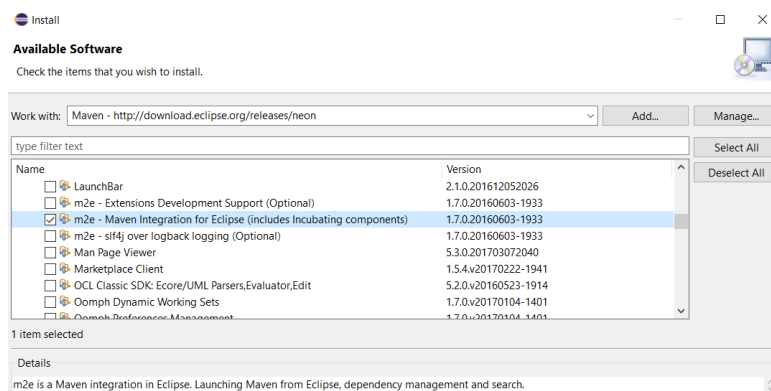
Part 1: Install Maven for Eclipse.

Most Eclipse downloads include the maven tooling already. If this is missing in your installation, follow these steps, else jump to part two:

1. Open the plugin installation window by selecting **“Help → Install new software”**.
2. Click on Add and type **“Maven”** for the name and <http://download.eclipse.org/releases/neon> for the location.



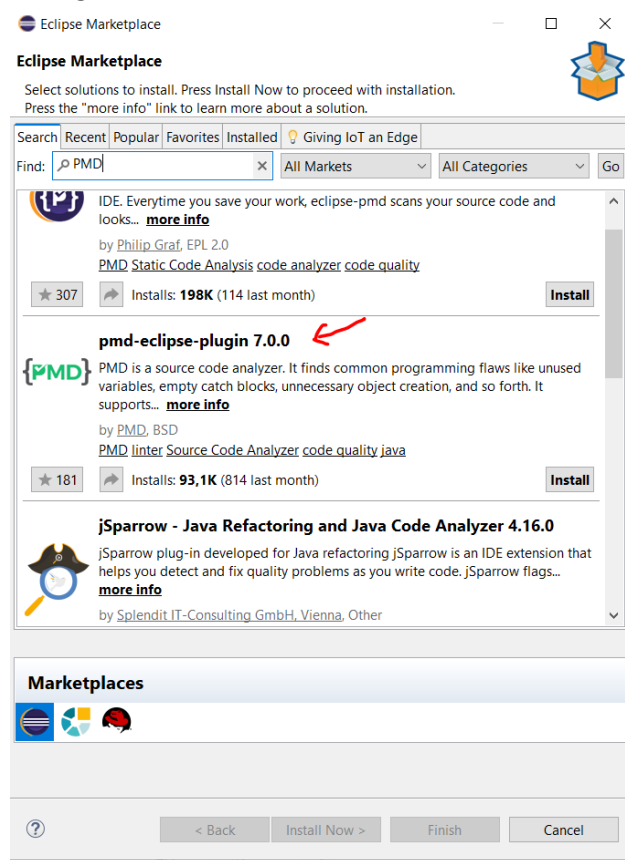
3. Click **“Add”** again and wait until the process finishes.
4. Search Maven and check **“Maven Integration for Eclipse”** under **“General Purpose Tools”**



5. Click **“Next”** and step through the following installation screens.

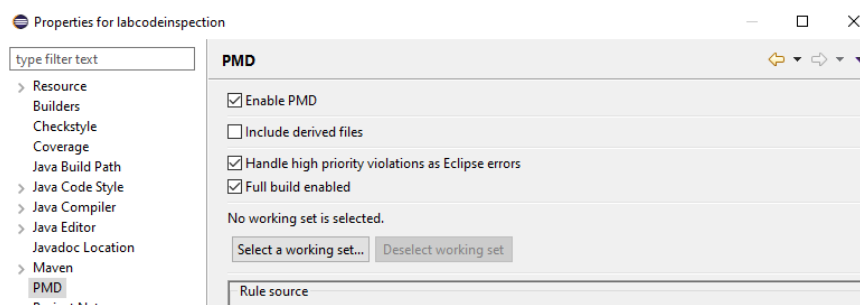
Part 2: Install the Eclipse PDM plug-in

1. Open the plugin installation window by selecting the “**Help → Marketplace**”.
2. Search for the PMD plugin
3. Install the selected Plugin



Part 3: Download and configure the project.

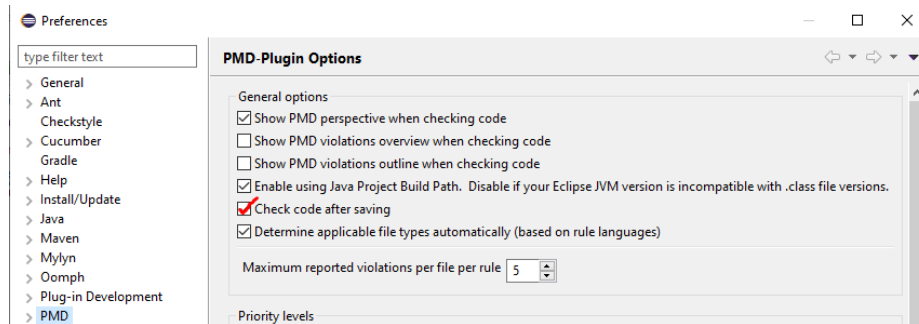
1. Clone or download the following repository:
<https://github.com/leortyz/CodeInspection.git> and open the project in eclipse.
Make sure you use your own repository after cloning it.
2. PMD will not be activated for the project by default. Open the project properties window by clicking in “**Project → 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

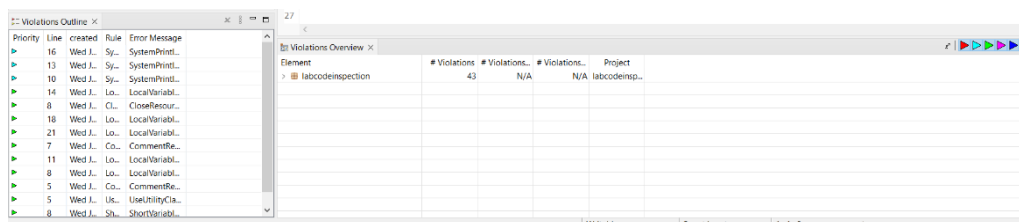
Go to “**Window → Preferences**”, select “**PMD**” and check “**Check code after saving**”



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.
- and **Blue** is Warning → Low priority.



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

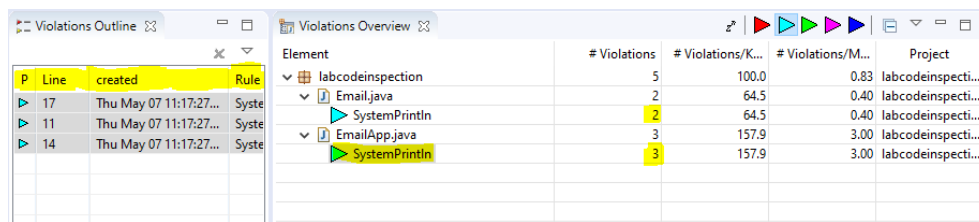
PMD shows the violations next to the lines that generate them.



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 EmailApp.java have 2 and 3 violations respectively for the rule “**SystemPrintln**”.

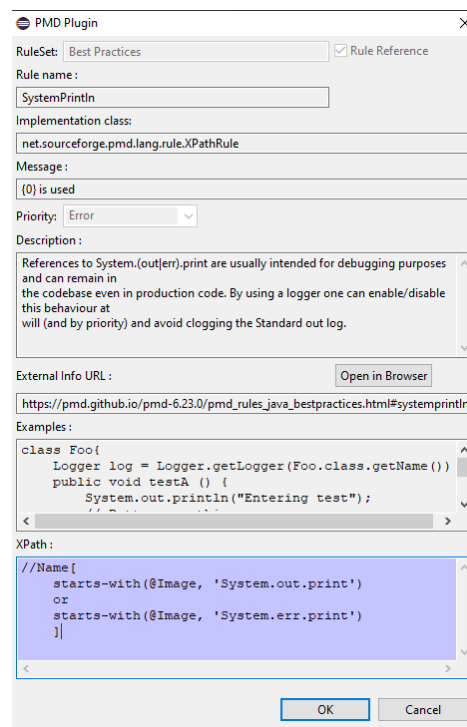
If we double-click on an element, the Violation Outline window will update showing all errors related to a file and some important data such as the violation line, the affected rule, and the error message.



Line	created	Rule
17	Thu May 07 11:17:27...	Syste
11	Thu May 07 11:17:27...	Syste
14	Thu May 07 11:17:27...	Syste

Element	# Violations	# Violations/K...	# Violations/M...	Project
labcodeinspection	5	100.0	0.83	labcodeinspecti...
Email.java	2	64.5	0.40	labcodeinspecti...
SystemPrintln	2	64.5	0.40	labcodeinspecti...
EmailApp.java	3	157.9	3.00	labcodeinspecti...
SystemPrintln	3	157.9	3.00	labcodeinspecti...

At “Violations Outline” tab, we can right-click on a violation and select "Show details..." for more information and an example solution.



PMD Plugin

RuleSet: Best Practices ☒ Rule Reference

Rule name: SystemPrintln

Implementation class: net.sourceforge.pmd.lang.rule.XPathRule

Message: {0} is used

Priority: Error

Description: References to System.out.print are usually intended for debugging purposes and can remain in the codebase even in production code. By using a logger one can enable/disable this behaviour at will (and by priority) and avoid clogging the Standard out log.

External Info URL: https://pmd.github.io/pmd-6.23.0/pmd_rules_java_bestpractices.html#systemprintln [Open in Browser](#)

Examples:

```
class Foo{
    Logger log = Logger.getLogger(Foo.class.getName())
    public void testA () {
        System.out.println("Entering test");
    }
}
```

XPath:

```
//Name [
    starts-with(@Image, 'System.out.print')
    or
    starts-with(@Image, 'System.err.print')
]
```

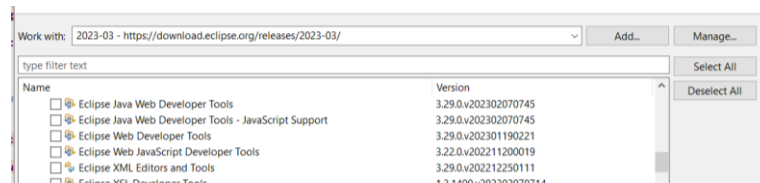
OK Cancel

For more information about configurations, refer to section 2 and 3 from chapter 22 of the book [2] and PMD website [1].

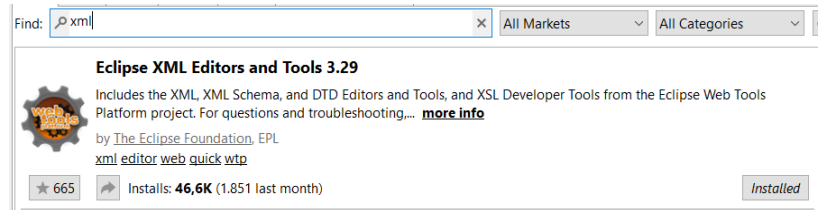
Part 5: Creating PMD rulesets

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:

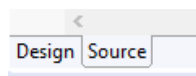
1. Right click on the project, then **New → Other → XML → XML File**.
2. If the option does not appear, install it in **help → Install new software**, with **Eclipse XML Editors and Tools** in the section **Web, XML, Java EE and OSGi Enterprise Development**.



And in the marketplace search for XML Editors and Tools



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



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

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

<ruleset name="<your name> 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
    http://pmd.sourceforge.io/ruleset_2_0_0.xsd">

  <description>
    Code Inspection Lab, <your full name>
  </description>

</ruleset>
```

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

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

This ruleset comes by default with PMD and it has rules that flag suboptimal code.

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

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

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

8. 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>
```

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

- Preferences**

type filter text

 - > General
 - > Ant
 - Checkstyle
 - > Cucumber
 - Gradle
 - > Help
 - > Install/Update
 - > Java
 - > Maven
 - > Mylyn
 - > Oomph
 - > Plug-in Development
 - ▼ PMD
 - CPD Preferences
 - File Filters
 - Reports
 - Rule Configuration**
 - > Run/Debug
 - > Team
 - Validation
 - > XML

Rule Configuration

☒ Use global rule management

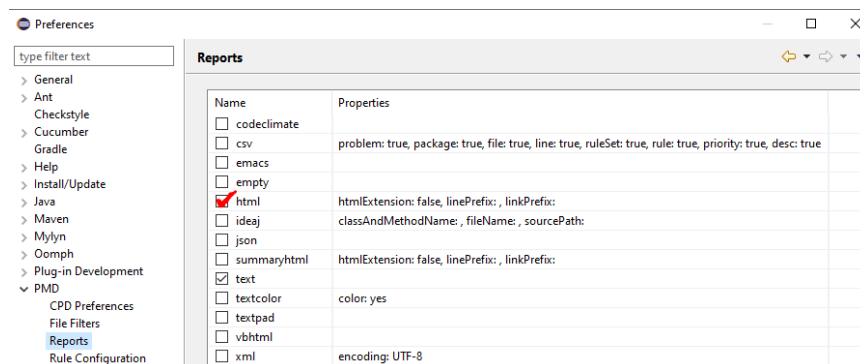
If global rule management is enabled, you can deactivate rules here globally. This is useful in order to ignore some rules temporarily. This setting overrides project-specific settings.

Rules grouped by: **<no grouping>** Active rules: 81 / 81

Rule	Severity	Score	Rule set	Type	M	A
<input checked="" type="checkbox"/> AbstractClassWithoutAbstractMethod	Green	3.0	Best Practices	T		
<input checked="" type="checkbox"/> AccessorClassGeneration	Green	1.04	Best Practices	T		
<input checked="" type="checkbox"/> AccessorMethodGeneration	Green	5.5.4	Best Practices	T		
<input checked="" type="checkbox"/> AddEmptyString	Green	4.0	Performance	X, T		
<input checked="" type="checkbox"/> AppendCharacterWithChar	Green	3.5	Performance	T		
<input checked="" type="checkbox"/> ArraysStoredDirectly	Green	2.2	Best Practices	T		
<input checked="" type="checkbox"/> AvoidArrayLoops	Green	3.5	Performance	X, T		
<input checked="" type="checkbox"/> AvoidFileStream	Red	6.0.0	Performance	X, T		Lar
<input checked="" type="checkbox"/> AvoidInstantiatingObjectsInLoops	Green	2.2	Performance	T		
<input checked="" type="checkbox"/> AvoidMessageDialogField	Green	6.18.0	Best Practices	X, T		

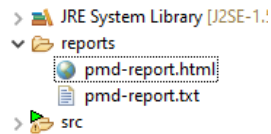
- Note that it is possible to configure the properties for each rule we add. To learn more about rulesets and their properties, refer to PMD Java Rules [3]

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.

```

3 public class Email {
4
5     private String firstName; //NOPMD This field will be manipulated later.
6     private String lastName;
7     private String password = null;
8     private String department;
9     private int defaultpasswordLength = 8;
10    private String email;
11

```

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

```

5 @SuppressWarnings("PMD.UseUtilityClass")
6 public class EmailApp {

```

Please note that only that rule will be ignored.

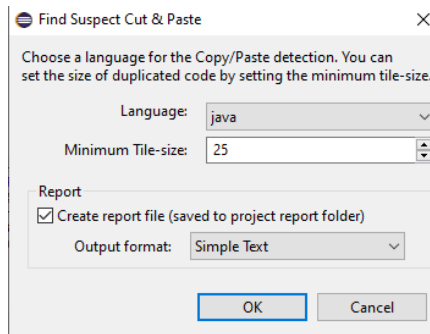
6. Save the file and see the results.

For more information about suppressing rules, check the PMD website [4] and section 7 from chapter 22 of the book [2]

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.

1. Just for demonstration purpose, copy and paste the “**randomPassword**” method from Email.java to EmailApp.java.
2. 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.

```

1 Found a 11 line (74 tokens) duplication in the following files:
2 Starting at line 33 of C:\Users\josea\OneDrive\Documents\EclipseProjects\labcodeinspection\src\main\java\labcodeinspection\Email.java
3 Starting at line 26 of C:\Users\josea\OneDrive\Documents\EclipseProjects\labcodeinspection\src\main\java\labcodeinspection\EmailApp.java
4
5 }
6
7 private String randomPassword(int length) {
8     String set = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890#$%&*";
9     char[] password = new char[length];
10    for (int i = 0; i < length; i++) {
11        int rand = (int) (Math.random() * set.length());
12        password[i] = set.charAt(rand);
13    }
14    return new String(password);
15 }

```

5. Revert the changes made in EmailApp.java.

Part 9: PMD and Maven

1. Open "pom.xml" file

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6     <groupId>com.espol.edu</groupId>
7     <artifactId>labcodeinspection</artifactId>
8     <version>0.0.1-SNAPSHOT</version>
9     <name>Lab Code Inspection</name>
10    <description>Software Engineering II</description>
11 </project>

```

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

```

<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>

```

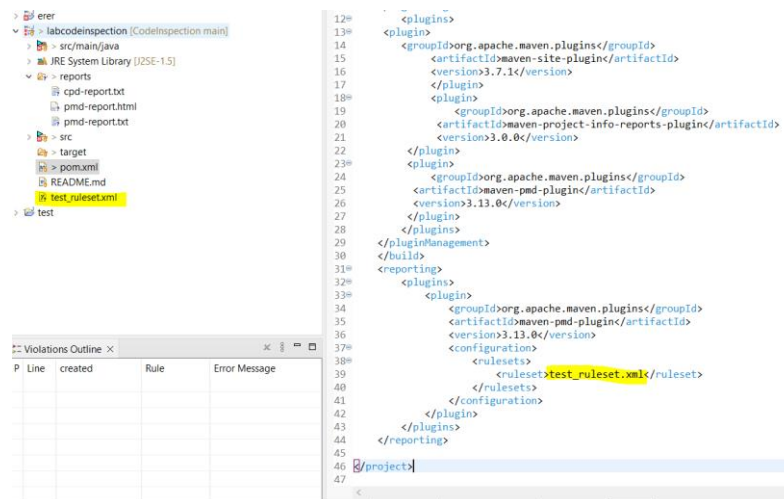
```

<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pmd-plugin</artifactId>
      <version>3.13.0</version>
      <configuration>
        <rulesets>
          <ruleset><yourname>_ruleset.xml</ruleset>
        </rulesets>
      </configuration>
    </plugin>
  </plugins>
</reporting>

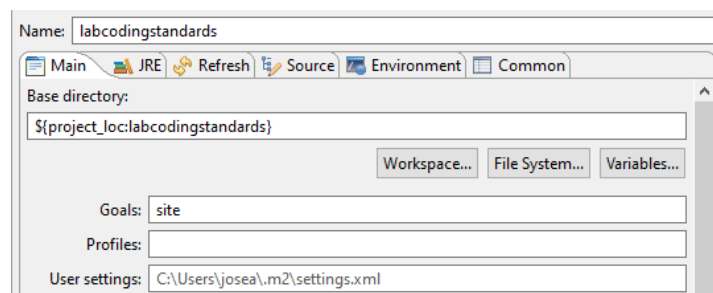
```

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.

Like this:



3. Save the file, right Click on it, then **“Run as → Maven build...”**
4. Type **“site”** for the Goals and Click Run.



5. Wait for the process to finish.

```

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 17.972 s
[INFO] Finished at: 2020-05-07T23:17:24-05:00
[INFO] -----

```

6. Go to your project directory, open **“target → site”**. This folder is visible from the tree project in Eclipse too.

7. Several html files are shown. Open “index.html”.


Lab Code Inspection

Last Published: 2020-05-07 | Version: 0.0.1-SNAPSHOT

Lab Code Inspection

Project Documentation

- Project Information
- Dependency Information
- About**
- Plugin Management
- Plugins
- Summary
- Project Reports

Built by: 

About Lab Code Inspection





Software Engineering II

Copyright © 2020.All rights reserved.

8. Click on “Project Reports → PMD” to see a detailed PMD report of your project.

Files

labcodeinspection/Email.java

Rule	Violation	Priority	Line
ImmutableField 	Private field 'lastName' could be made final; it is only initialized in the declaration or constructor.	3	6
RedundantFieldInitializer 	Avoid using redundant field initializer for 'password'	3	7
ImmutableField 	Private field 'defaultpasswordLength' could be made final; it is only initialized in the declaration or constructor.	3	9
SwitchStmtsShouldHaveDefault 	Switch statements should have a default label	3	23 -32

Copyright © 2020.All rights reserved.

Development

1. Delete all comments and annotations from the code.
2. Add the following ruleset and rules:
 - ClassNamingConventions
 - Rule “**BeanMembersShouldSerialize**” from Error Prone ruleset with a priority of 2
 - Rule “**UseLocaleWithCaseConversions**” 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

3. Generate an html report, make a copy and save it somewhere on your disk.
4. Correct any violation in the code that have been generated in the report.
5. Generate a new report (without violations).

Challenge

- Create a branch in the repository of the previous workshop, CODING STANDARDS - I TERM 2023, and add the new requirements for the vacation system that are at the end of this document.
- Configure PMD for that project and generate its respective report.
- Create your **own configuration file** for this case and add the rules you think are necessary, minimum 4.
- Correct any violation in the code that have been generated in the report.
- Generate a new report (without violations).

Deliverables

1. Lab report with screenshots of the process.
2. Two PMD reports.
3. Challenge
4. Your own xml configuration file for the challenge
5. Include in the report the **url** of the repository where you performed the lab.

Rubric

Description	Value
Lab report Criteria: lab report content (30 pts) <ul style="list-style-type: none">All the evidences: screenshots of the process, initial and fixed reports, challenge and url are included: 30 ptsJust both reports are included: 15 ptsEmpty, partial or undelivered report: 0 pts Criteria: fixed issues (20 pts) <ul style="list-style-type: none">All the violations were fixed: 20 ptsSome violations were fixed: 5 ptsEmpty report or neither violation fixed: 0 pts Criteria: support (10 pts) <ul style="list-style-type: none">The fixed PMD report should be in concordance with the source-code (repository): 10 ptsOtherwise: 0 pts	60
Challenge Criteria: Code generation (20 pts) <ul style="list-style-type: none">The branch fulfills the new requirements of the system (20 pts)There is not addition of the new requirements (0 pts) Criteria: PMD tool (20 pts) <ul style="list-style-type: none">There is evidence of the PMD tool use, and the fixes are corrected (20 pts)There is no report (0 pts)	40
Penalty for not having url of a public repository	-100

Late Submission Policy

Delay (§)	Penalty (Ω)
1 hour or less	loss of 10%
1 to 6 hours	loss of 20%
6 to 24 hours	loss of 30%
Over 24 hours:	loss of 100%

(§) every clock hour counts including weekends or holidays.

(Ω) automatic and non-negotiable penalty

References

- [1] PMD, "PMD Source Code Analyzer," [Online]. Available: <https://pmd.github.io/>.
- [2] J. Ferguson, Java Power Tools, O'Reilly Media, 2008.
- [3] PMD, "Java Rules," [Online]. Available: https://pmd.github.io/latest/pmd_rules_java.html.
- [4] PMD, "Suppressing warnings," [Online]. Available: https://pmd.github.io/latest/pmd_userdocs_suppressing_warnings.html.
- [5] Instructions, <https://github.com/leortyz/softwareEngineeringResources/wiki/Code-Inspection>

Vacation Package Cost Estimator

User Requirements

- The system should be able to calculate the cost of vacation packages based on the following information:
 - Destination of the vacation
 - Number of travelers
 - Duration of the vacation in days
- The base cost for a vacation package is \$1000, and it should be applied to every package.
- If the destination is a popular tourist spot, an additional cost based on the destination should be added to the base cost. The popular tourist spots are:
 - Paris - \$500
 - New York City - \$600
- If the number of travelers is more than 4, but less than 10, a group discount of 10% should be applied to the total cost.
- If the number of travelers is more than 10, a group discount of 20% should be applied to the total cost.
- If the duration of the vacation is less than 7 days, a penalty fee of \$200 should be added to the total cost.
- If the duration of the vacation is more than 30 days or the number of passengers is 2, \$200 must be subtracted from the total cost, as it is a promotion policy.
- The vacation package is not available for groups of more than 80 persons.
- The system should provide the following outputs:
 - The total cost of the vacation package (a positive integer) if all the input data is valid.
 - A value of -1 if the input data is not valid.
- The system should validate the input entered by the users to ensure data accuracy.
- The system should handle errors gracefully and provide clear error messages if the user enters invalid input or if there are any calculation errors.

For this new version of the system, the client has added some requirements that will increment the user experience.

- The system should allow the user to select optional add-ons to customize their vacation package. The available add-ons are:
 - All-Inclusive Package - \$200 per traveler
 - Adventure Activities Package - \$150 per traveler
 - Spa and Wellness Package - \$100 per traveler
- The system should calculate the cost of the selected add-ons and include it in the total cost of the vacation package.