Ant Task Usage

## **PMD**

### **Description**

Runs a set of static code analysis rules on some source code files and generates a list of problems found.

### **Installation**

First you need to download PMD’s binary distribution zip file. Then you can either copy all “\*.jar” files from PMD’s lib folder into one of ANT’s library folders (ANT\_HOME/lib, ${user.home}/.ant/lib) or using the -lib command line parameter.

However, the preferred way is to define a <classpath> for pmd itself and use this classpath when adding the PMD Task. Assuming, you have extracted the PMD zip file to /home/joe/pmd-bin-6.31.0, then you can make use of the PMD Task like this:

<taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask">

<classpath>

<fileset dir="/home/joe/pmd-bin-6.31.0/lib">

<include name="\*.jar"/>

</fileset>

</classpath>

</taskdef>

Alternatively, a path can be defined and used via classpathref

<path id="pmd.classpath">

<fileset dir="/home/joe/pmd-bin-6.31.0/lib">

<include name="\*.jar"/>

</fileset>

</path>

<taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask" classpathref="pmd.classpath" />

The examples below won’t repeat this taskdef element, as this is always required.

### **Parameters**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Required** |
| rulesetfiles | A comma delimited list of ruleset files ('rulesets/java/quickstart.xml,config/my-ruleset.xml'). If you write your own ruleset files, you can put them on the classpath and plug them in here. | Yes, unless the ruleset nested element is used |
| failonerror | Whether or not to fail the build if any errors occur while processing the files | No |
| failOnRuleViolation | Whether or not to fail the build if PMD finds any problems | No |
| minimumPriority | The rule priority threshold; rules with lower priority than they will not be used | No |
| shortFilenames | Places truncated filenames in the report. This can reduce your report file size by 15%-20%. | No |
| failuresPropertyName | A property name to plug the number of rule violations into when the task finishes | No |
| encoding | The character set encoding (e.g. UTF-8) to use when reading the source code files | No |
| suppressMarker | The series of characters to use to tell PMD to skip lines - the default is NOPMD. | No |
| maxRuleViolations | Whether or not to fail the build if PMD finds more than the value of this attribute. Note that setting this attribute does not require to set the failOnRuleViolation to true. | No |
| cacheLocation | The location of the analysis cache file to be used. Setting this property enables Incremental Analysis, which can greatly improve analysis time without loosing analysis quality. **Its use is strongly recommended.** | No |
| noCache | Setting this property to true disables Incremental Analysis, even if *cacheLocation* is provided. You can use this to explicitly turn off suggestions to use incremental analysis, or for testing purposes. | No |

formatter nested element - specifies the format of and the files to which the report is written. You can configure multiple formatters.

|  |  |
| --- | --- |
| **Name** | **Values** |
| type | xml,ideaj,textcolor,text,textpad,emacs,csv,html,xslt,yahtml,summaryhtml,vbhtml,codeclimate |
| showSuppressed | Whether to show suppressed warnings; "false" is the default. |
| toFile | A filename to which to write the report |
| toConsole | Whether to output the report to the console; "false" is the default. |
| The formatter element can contain nested param elements to configure the formatter in detail, e.g.  **encoding**  Specifies the encoding to be used in the generated report (only honored when used with `toFile`). When rendering `toConsole` PMD will automatically detect the terminal's encoding and use it, unless the output is being redirected / piped, in which case `file.encoding` is used. See example below.  **linkPrefix**  Used for linking to online HTMLized source (like [this](https://maven.apache.org/plugins/maven-pmd-plugin/xref/org/apache/maven/plugins/pmd/PmdReport.html)). See example below. Note, this only works with [maven-jxr-plugin](https://maven.apache.org/jxr/maven-jxr-plugin/index.html).  **linePrefix**  Used for linking to online HTMLized source (like [this](https://maven.apache.org/plugins/maven-pmd-plugin/xref/org/apache/maven/plugins/pmd/PmdReport.html#L375)). See example below. Note, this only works with [maven-jxr-plugin](https://maven.apache.org/jxr/maven-jxr-plugin/index.html). | |

sourceLanguage nested element - specify which language (Java, Ecmascript, XML,…) and the associated version (1.5, 1.6,…). This element is optional. The language is determined by file extension automatically and the latest language version is used.

ruleset nested element - another way to specify rulesets. You can specify multiple elements. Here’s an example:

<target name="pmd">

<taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>

<pmd shortFilenames="true">

<ruleset>rulesets/java/quickstart.xml</ruleset>

<ruleset>config/my-ruleset.xml</ruleset>

<fileset dir="/usr/local/j2sdk1.4.1\_01/src/">

<include name="java/lang/\*.java"/>

</fileset>

</pmd>

</target>

### **Language version selection**

PMD selects the language automatically using the file extension. If multiple versions of a language are supported, PMD uses the latest version as default. This is currently the case for Java only, which has support for multiple versions.

If a languages supports multiple versions, you can select a specific version here, so that e.g. rules, that only apply to specific versions, are not executed. E.g. the rule [UseTryWithResources](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#usetrywithresources) only makes sense with Java 1.7 and later. If your project uses Java 1.5, then you should configure the sourceLanguage accordingly and this rule won’t be executed.

The specific version of a language to be used is selected via the sourceLanguage nested element. Possible values are:

<sourceLanguage name="java" version="1.8"/>

### **Examples**

#### **One ruleset**

Running one ruleset to produce a HTML report (and printing the report to the console as well) using a file cache

<target name="pmd">

<pmd rulesetfiles="rulesets/java/quickstart.xml" cacheLocation="build/pmd/pmd.cache">

<formatter type="html" toFile="pmd\_report.html" toConsole="true"/>

<fileset dir="C:\j2sdk1.4.1\_01\src\java\lang\">

<include name="\*\*/\*.java"/>

</fileset>

</pmd>

</target>

#### **Multiple rulesets**

Running multiple rulesets to produce an XML report with the same analysis cache

<target name="pmd">

<pmd rulesetfiles="rulesets/java/quickstart.xml,config/my-ruleset.xml" cacheLocation="build/pmd/pmd.cache">

<formatter type="xml" toFile="c:\pmd\_report.xml"/>

<fileset dir="C:\j2sdk1.4.1\_01\src\java\lang\">

<include name="\*\*/\*.java"/>

</fileset>

</pmd>

</target>

#### **Full example with auxclasspath**

<project name="MyProject" default="pmd" basedir=".">

<property name="src" location="src"/>

<property name="build" location="build"/>

<path id="project.dependencies">

<pathelement location="lib/third-party.jar"/>

<pathelement location="lib/xyz.jar"/>

</path>

<path id="pmd.classpath">

<fileset dir="/home/joe/pmd-bin-6.31.0/lib">

<include name="\*.jar"/>

</fileset>

</path>

<taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask" classpathref="pmd.classpath" />

<target name="init">

<mkdir dir="${build}"/>

</target>

<target name="compile" depends="init">

<javac srcdir="${src}" destdir="${build}" classpathref="project.dependencies"

source="1.8" target="1.8" />

</target>

<target name="pmd" depends="compile">

<pmd cacheLocation="${build}/pmd.cache">

<auxclasspath>

<pathelement location="${build}"/>

<path refid="project.dependencies"/>

</auxclasspath>

<ruleset>rulesets/java/quickstart.xml</ruleset>

<formatter type="html" toFile="${build}/pmd\_report.html"/>

<sourceLanguage name="java" version="1.8"/>

<fileset dir="${src}">

<include name="\*\*/\*.java"/>

</fileset>

</pmd>

</target>

<target name="clean">

<delete dir="${build}"/>

</target>

</project>

Working Example

<project name="demo" default="pmd" basedir=".">

<property name="build.dir" value="build"/>

<property name="src.dir" value="src"/>

<!-- Lib for PMD -->

<path id="pmd.classpath">

<fileset dir="C:\software\pmd-bin-6.31.0\lib">

<include name="\*.jar"/>

</fileset>

</path>

<!-- Task for PMD-->

<taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask" classpathref="pmd.classpath">

</taskdef>

<!-- PMD Target -->

<target name="pmd">

<pmd shortFilenames="true" failOnRuleViolation="true">

<ruleset>${src.dir}/fis\_rules.xml</ruleset>

<formatter type="summaryhtml" toFile="${build.dir}/pmd\_report.html"/>

<sourceLanguage name="java" version="1.8"/>

<fileset dir="..\VCPP\_Services\${src.dir}">

<include name="\*\*/\*.java"/>

</fileset>

<fileset dir="${src.dir}">

<include name="\*\*/\*.java"/>

</fileset>

</pmd>

</target>

</project>

XML Rule Set

<?xml version=*"1.0"*?>

<ruleset name=*"quickstart"* 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>Quickstart configuration of PMD. Includes the rules that are most likely to apply everywhere.</description>

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

ref="category/java/codestyle.xml/DuplicateImports"/> <rule ref="category/java/bestpractices.xml/UnusedImports"/> -->

<rule ref=*"category/java/bestpractices.xml/UnusedLocalVariable"* />

<rule ref=*"category/java/bestpractices.xml/UnusedPrivateField"* />

<rule ref=*"category/java/bestpractices.xml/UnusedPrivateMethod"* />

</ruleset>

# Making rulesets

A ruleset is an XML configuration file, which describes a collection of rules to be executed in a PMD run. PMD includes built-in rulesets to run quick analyses with a default configuration, but users are encouraged to make their own rulesets from the start, because they allow for so much configurability. This page walk you through the creation of a ruleset and the multiple configuration features offered by rulesets.

## **Creating a ruleset**

The first step is to create a new empty ruleset. You can use the following template:

**<?xml version="1.0"?>**

<ruleset name="Custom 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>

My custom rules

</description>

*<!-- Your rules will come here -->*

</ruleset>

### **Referencing a single rule**

To use the built-in rules PMD provides, you need to add some references to them. Here’s a basic rule reference:

<rule ref="category/java/errorprone.xml/EmptyCatchBlock" />

Adding that element into the ruleset element adds the rule [EmptyCatchBlock](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptycatchblock) to your ruleset. This is a Java rule, so it will be executed on every Java file PMD encounters in its search space.

How to read the ref attribute?

* category/java/errorprone.xml is a reference to the Java category errorprone. Since PMD 6.0.0, all PMD built-in rules are sorted in one of eight categories, which are consistent across languages:
  1. **Best Practices**: These are rules which enforce generally accepted best practices.
  2. **Code Style**: These rules enforce a specific coding style.
  3. **Design**: Rules that help you discover design issues.
  4. **Documentation**: These rules are related to code documentation.
  5. **Error Prone**: Rules to detect constructs that are either broken, extremely confusing or prone to runtime errors.
  6. **Multithreading**: These are rules that flag issues when dealing with multiple threads of execution.
  7. **Performance**: Rules that flag suboptimal code.
  8. **Security**: Rules that flag potential security flaws.”
* EmptyCatchBlock is simply the name of the rule. If there were no rule with that name within the specified category, then PMD would fail before starting the analysis.

### **Bulk-adding rules**

You can also reference rules in bulk by referencing a complete category or ruleset, possibly excluding certain rules, like in the following:

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

<exclude name="WhileLoopsMustUseBraces"/>

<exclude name="IfElseStmtsMustUseBraces"/>

</rule>

Here, the ref attribute references a whole category. You can also use a file system path or classpath relative path. In any case, the path must address an accessible ruleset XML file.

### **Filtering the processed files**

You can exclude some files from being processed by a ruleset using **exclude patterns**, with an optional overridding **include pattern**. A file will be excluded from processing when there is a matching exclude pattern, but no matching include pattern. This exclude/include technique works regardless of how PMD is used (e.g. command line, IDE, Ant), making it easier to keep application of your PMD rules consistent throughout your environment. Here is an example:

**<?xml version="1.0"?>**

<ruleset name="myruleset"

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>My ruleset</description>

<exclude-pattern>.\*/some/package/.\*</exclude-pattern>

<exclude-pattern>.\*/some/other/package/FunkyClassNamePrefix.\*</exclude-pattern>

<include-pattern>.\*/some/package/ButNotThisClass.\*</include-pattern>

*<!-- Rules here ... -->*

</ruleset>

# Configuring rules

## **Message and priority overriding**

You can change a rule’s **message** by specifying a message attribute on the rule element. This will override the previous value and change the message the rule will print on the report.

Similarly, the **priority** of a rule can be changed via a nested element. Using priority, you can deactivate some rules based on a minimum priority threshold (set using the -min CLI option). Priority is an integer ranging from 1 to 5, with 1 being the highest priority.

Putting things together, the following rule reference lowers the priority of EmptyCatchBlock to 5, such that e.g. using the -min 4 CLI parameters will cause the rule to be ignored.

<rule ref="category/java/errorprone.xml/EmptyCatchBlock"

message="Empty catch blocks should be avoided" >

<priority>5</priority>

</rule>

## **Best Practices**

Rules which enforce generally accepted best practices.

* [AbstractClassWithoutAbstractMethod](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#abstractclasswithoutabstractmethod): The abstract class does not contain any abstract methods. An abstract class suggests an incomplet…
* [AccessorClassGeneration](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#accessorclassgeneration): Instantiation by way of private constructors from outside of the constructor’s class often causes…
* [AccessorMethodGeneration](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#accessormethodgeneration): When accessing private fields / methods from another class, the Java compiler will generate acces…
* [ArrayIsStoredDirectly](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#arrayisstoreddirectly): Constructors and methods receiving arrays should clone objects and store the copy. This prevents …
* [AvoidMessageDigestField](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#avoidmessagedigestfield): Declaring a MessageDigest instance as a field make this instance directly available to multiple t…
* [AvoidPrintStackTrace](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#avoidprintstacktrace): Avoid printStackTrace(); use a logger call instead.
* [AvoidReassigningCatchVariables](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#avoidreassigningcatchvariables): Reassigning exception variables caught in a catch statement should be avoided because of: 1) If i…
* [AvoidReassigningLoopVariables](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#avoidreassigningloopvariables): Reassigning loop variables can lead to hard-to-find bugs. Prevent or limit how these variables ca…
* [AvoidReassigningParameters](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#avoidreassigningparameters): Reassigning values to incoming parameters is not recommended. Use temporary local variables inst…
* [AvoidStringBufferField](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#avoidstringbufferfield): StringBuffers/StringBuilders can grow considerably, and so may become a source of memory leaks if…
* [AvoidUsingHardCodedIP](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#avoidusinghardcodedip): Application with hard-coded IP addresses can become impossible to deploy in some cases. Externali…
* [CheckResultSet](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#checkresultset): Always check the return values of navigation methods (next, previous, first, last) of a ResultSet…
* [ConstantsInInterface](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#constantsininterface): Avoid constants in interfaces. Interfaces should define types, constants are implementation detai…
* [DefaultLabelNotLastInSwitchStmt](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#defaultlabelnotlastinswitchstmt): By convention, the default label should be the last label in a switch statement.
* [DoubleBraceInitialization](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#doublebraceinitialization): Double brace initialisation is a pattern to initialise eg collections concisely. But it implicitl…
* [ForLoopCanBeForeach](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#forloopcanbeforeach): Reports loops that can be safely replaced with the foreach syntax. The rule considers loops over …
* [ForLoopVariableCount](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#forloopvariablecount): Having a lot of control variables in a ‘for’ loop makes it harder to see what range of values the…
* [GuardLogStatement](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#guardlogstatement): Whenever using a log level, one should check if the loglevel is actually enabled, or otherwise sk…
* [JUnit4SuitesShouldUseSuiteAnnotation](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#junit4suitesshouldusesuiteannotation): In JUnit 3, test suites are indicated by the suite() method. In JUnit 4, suites are indicated thr…
* [JUnit4TestShouldUseAfterAnnotation](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#junit4testshoulduseafterannotation): In JUnit 3, the tearDown method was used to clean up all data entities required in running tests….
* [JUnit4TestShouldUseBeforeAnnotation](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#junit4testshouldusebeforeannotation): In JUnit 3, the setUp method was used to set up all data entities required in running tests. JUni…
* [JUnit4TestShouldUseTestAnnotation](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#junit4testshouldusetestannotation): In JUnit 3, the framework executed all methods which started with the word test as a unit test. I…
* [JUnitAssertionsShouldIncludeMessage](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#junitassertionsshouldincludemessage): JUnit assertions should include an informative message - i.e., use the three-argument version of …
* [JUnitTestContainsTooManyAsserts](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#junittestcontainstoomanyasserts): Unit tests should not contain too many asserts. Many asserts are indicative of a complex test, fo…
* [JUnitTestsShouldIncludeAssert](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#junittestsshouldincludeassert): JUnit tests should include at least one assertion. This makes the tests more robust, and using a…
* [JUnitUseExpected](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#junituseexpected): In JUnit4, use the @Test(expected) annotation to denote tests that should throw exceptions.
* [LiteralsFirstInComparisons](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#literalsfirstincomparisons): Position literals first in all String comparisons, if the second argument is null then NullPointe…
* [LooseCoupling](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#loosecoupling): The use of implementation types (i.e., HashSet) as object references limits your ability to use a…
* [MethodReturnsInternalArray](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#methodreturnsinternalarray): Exposing internal arrays to the caller violates object encapsulation since elements can be remove…
* [MissingOverride](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#missingoverride): Annotating overridden methods with @Override ensures at compile time that the method …
* [OneDeclarationPerLine](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#onedeclarationperline): Java allows the use of several variables declaration of the same type on one line. However, it ca…
* [PositionLiteralsFirstInCaseInsensitiveComparisons](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#positionliteralsfirstincaseinsensitivecomparisons): Deprecated Position literals first in comparisons, if the second argument is null then NullPointerExceptions…
* [PositionLiteralsFirstInComparisons](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#positionliteralsfirstincomparisons): Deprecated Position literals first in comparisons, if the second argument is null then NullPointerExceptions…
* [PreserveStackTrace](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#preservestacktrace): Throwing a new exception from a catch block without passing the original exception into the new e…
* [ReplaceEnumerationWithIterator](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#replaceenumerationwithiterator): Consider replacing Enumeration usages with the newer java.util.Iterator
* [ReplaceHashtableWithMap](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#replacehashtablewithmap): Consider replacing Hashtable usage with the newer java.util.Map if thread safety is not required.
* [ReplaceVectorWithList](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#replacevectorwithlist): Consider replacing Vector usages with the newer java.util.ArrayList if expensive thread-safe oper…
* [SwitchStmtsShouldHaveDefault](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#switchstmtsshouldhavedefault): All switch statements should include a default option to catch any unspecified values.
* [SystemPrintln](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#systemprintln): References to System.(out|err).print are usually intended for debugging purposes and can remain …
* [UnusedAssignment](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#unusedassignment): Reports assignments to variables that are never used before the variable is overwritten, …
* [UnusedFormalParameter](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#unusedformalparameter): Reports parameters of methods and constructors that are not referenced them in the method body. P…
* [UnusedImports](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#unusedimports): Avoid unused import statements to prevent unwanted dependencies. This rule will also find unused …
* [UnusedLocalVariable](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#unusedlocalvariable): Detects when a local variable is declared and/or assigned, but not used. Variables whose name sta…
* [UnusedPrivateField](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#unusedprivatefield): Detects when a private field is declared and/or assigned a value, but not used.
* [UnusedPrivateMethod](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#unusedprivatemethod): Unused Private Method detects when a private method is declared but is unused.
* [UseAssertEqualsInsteadOfAssertTrue](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#useassertequalsinsteadofasserttrue): This rule detects JUnit assertions in object equality. These assertions should be made by more sp…
* [UseAssertNullInsteadOfAssertTrue](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#useassertnullinsteadofasserttrue): This rule detects JUnit assertions in object references equality. These assertions should be made…
* [UseAssertSameInsteadOfAssertTrue](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#useassertsameinsteadofasserttrue): This rule detects JUnit assertions in object references equality. These assertions should be made…
* [UseAssertTrueInsteadOfAssertEquals](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#useasserttrueinsteadofassertequals): When asserting a value is the same as a literal or Boxed boolean, use assertTrue/assertFalse, ins…
* [UseCollectionIsEmpty](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#usecollectionisempty): The isEmpty() method on java.util.Collection is provided to determine if a collection has any ele…
* [UseTryWithResources](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#usetrywithresources): Java 7 introduced the try-with-resources statement. This statement ensures that each resource is …
* [UseVarargs](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#usevarargs): Java 5 introduced the varargs parameter declaration for methods and constructors. This syntactic…
* [WhileLoopWithLiteralBoolean](https://pmd.github.io/latest/pmd_rules_java_bestpractices.html#whileloopwithliteralboolean): ‘do {} while (true);’ requires reading the end of the statement before it is apparent that it loo…

## **Code Style**

Rules which enforce a specific coding style.

* [AbstractNaming](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#abstractnaming): Deprecated Abstract classes should be named ‘AbstractXXX’. This rule is deprecated and will be removed with …
* [AtLeastOneConstructor](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#atleastoneconstructor): Each non-static class should declare at least one constructor. Classes with solely static members…
* [AvoidDollarSigns](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#avoiddollarsigns): Avoid using dollar signs in variable/method/class/interface names.
* [AvoidFinalLocalVariable](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#avoidfinallocalvariable): Deprecated Avoid using final local variables, turn them into fields. Note that this is a controversial rule …
* [AvoidPrefixingMethodParameters](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#avoidprefixingmethodparameters): Deprecated Prefixing parameters by ‘in’ or ‘out’ pollutes the name of the parameters and reduces code readab…
* [AvoidProtectedFieldInFinalClass](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#avoidprotectedfieldinfinalclass): Do not use protected fields in final classes since they cannot be subclassed. Clarify your intent…
* [AvoidProtectedMethodInFinalClassNotExtending](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#avoidprotectedmethodinfinalclassnotextending): Do not use protected methods in most final classes since they cannot be subclassed. This should o…
* [AvoidUsingNativeCode](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#avoidusingnativecode): Unnecessary reliance on Java Native Interface (JNI) calls directly reduces application portabilit…
* [BooleanGetMethodName](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#booleangetmethodname): Methods that return boolean results should be named as predicate statements to denote this. I.e, …
* [CallSuperInConstructor](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#callsuperinconstructor): It is a good practice to call super() in a constructor. If super() is not called but another cons…
* [ClassNamingConventions](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#classnamingconventions): Configurable naming conventions for type declarations. This rule reports type declara…
* [CommentDefaultAccessModifier](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#commentdefaultaccessmodifier): To avoid mistakes if we want that an Annotation, Class, Enum, Method, Constructor or Field have a…
* [ConfusingTernary](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#confusingternary): Avoid negation within an "if" expression with an "else" clause. For example, rephrase: ‘if (x !=…
* [ControlStatementBraces](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#controlstatementbraces): Enforce a policy for braces on control statements. It is recommended to use braces on ‘if … els…
* [DefaultPackage](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#defaultpackage): Use explicit scoping instead of accidental usage of default package private level. The rule allow…
* [DontImportJavaLang](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#dontimportjavalang): Avoid importing anything from the package ‘java.lang’. These classes are automatically imported …
* [DuplicateImports](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#duplicateimports): Duplicate or overlapping import statements should be avoided.
* [EmptyMethodInAbstractClassShouldBeAbstract](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#emptymethodinabstractclassshouldbeabstract): Empty or auto-generated methods in an abstract class should be tagged as abstract. This helps to …
* [ExtendsObject](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#extendsobject): No need to explicitly extend Object.
* [FieldDeclarationsShouldBeAtStartOfClass](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#fielddeclarationsshouldbeatstartofclass): Fields should be declared at the top of the class, before any method declarations, constructors, …
* [FieldNamingConventions](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#fieldnamingconventions): Configurable naming conventions for field declarations. This rule reports variable declarations …
* [ForLoopShouldBeWhileLoop](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#forloopshouldbewhileloop): Some for loops can be simplified to while loops, this makes them more concise.
* [ForLoopsMustUseBraces](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#forloopsmustusebraces): Deprecated Avoid using ‘for’ statements without using curly braces. If the code formatting or indentation is…
* [FormalParameterNamingConventions](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#formalparameternamingconventions): Configurable naming conventions for formal parameters of methods and lambdas. This ru…
* [GenericsNaming](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#genericsnaming): Names for references to generic values should be limited to a single uppercase letter.
* [IdenticalCatchBranches](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#identicalcatchbranches): Identical ‘catch’ branches use up vertical space and increase the complexity of code without …
* [IfElseStmtsMustUseBraces](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#ifelsestmtsmustusebraces): Deprecated Avoid using if..else statements without using surrounding braces. If the code formatting or inden…
* [IfStmtsMustUseBraces](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#ifstmtsmustusebraces): Deprecated Avoid using if statements without using braces to surround the code block. If the code formatting…
* [LinguisticNaming](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#linguisticnaming): This rule finds Linguistic Naming Antipatterns. It checks for fields, that are named, as if they …
* [LocalHomeNamingConvention](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#localhomenamingconvention): The Local Home interface of a Session EJB should be suffixed by ‘LocalHome’.
* [LocalInterfaceSessionNamingConvention](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#localinterfacesessionnamingconvention): The Local Interface of a Session EJB should be suffixed by ‘Local’.
* [LocalVariableCouldBeFinal](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#localvariablecouldbefinal): A local variable assigned only once can be declared final.
* [LocalVariableNamingConventions](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#localvariablenamingconventions): Configurable naming conventions for local variable declarations and other locally-scoped …
* [LongVariable](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#longvariable): Fields, formal arguments, or local variable names that are too long can make the code difficult t…
* [MDBAndSessionBeanNamingConvention](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#mdbandsessionbeannamingconvention): The EJB Specification states that any MessageDrivenBean or SessionBean should be suffixed by ‘Bean’.
* [MethodArgumentCouldBeFinal](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#methodargumentcouldbefinal): A method argument that is never re-assigned within the method can be declared final.
* [MethodNamingConventions](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#methodnamingconventions): Configurable naming conventions for method declarations. This rule reports method dec…
* [MIsLeadingVariableName](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#misleadingvariablename): Deprecated Detects when a non-field has a name starting with ‘m\_’. This usually denotes a field and could b…
* [NoPackage](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#nopackage): Detects when a class, interface, enum or annotation does not have a package definition.
* [OnlyOneReturn](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#onlyonereturn): A method should have only one exit point, and that should be the last statement in the method.
* [PackageCase](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#packagecase): Detects when a package definition contains uppercase characters.
* [PrematureDeclaration](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#prematuredeclaration): Checks for variables that are defined before they might be used. A reference is deemed to be prem…
* [RemoteInterfaceNamingConvention](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#remoteinterfacenamingconvention): Remote Interface of a Session EJB should not have a suffix.
* [RemoteSessionInterfaceNamingConvention](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#remotesessioninterfacenamingconvention): A Remote Home interface type of a Session EJB should be suffixed by ‘Home’.
* [ShortClassName](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#shortclassname): Short Classnames with fewer than e.g. five characters are not recommended.
* [ShortMethodName](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#shortmethodname): Method names that are very short are not helpful to the reader.
* [ShortVariable](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#shortvariable): Fields, local variables, or parameter names that are very short are not helpful to the reader.
* [SuspiciousConstantFieldName](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#suspiciousconstantfieldname): Deprecated Field names using all uppercase characters - Sun’s Java naming conventions indicating constants -…
* [TooManyStaticImports](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#toomanystaticimports): If you overuse the static import feature, it can make your program unreadable and unmaintainable,…
* [UnnecessaryAnnotationValueElement](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#unnecessaryannotationvalueelement): Avoid the use of value in annotations when it’s the only element.
* [UnnecessaryCast](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#unnecessarycast): This rule detects when a cast is unnecessary while accessing collection elements. This rule is mo…
* [UnnecessaryConstructor](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#unnecessaryconstructor): This rule detects when a constructor is not necessary; i.e., when there is only one constructor a…
* [UnnecessaryFullyQualifiedName](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#unnecessaryfullyqualifiedname): Import statements allow the use of non-fully qualified names. The use of a fully qualified name …
* [UnnecessaryLocalBeforeReturn](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#unnecessarylocalbeforereturn): Avoid the creation of unnecessary local variables
* [UnnecessaryModifier](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#unnecessarymodifier): Fields in interfaces and annotations are automatically ‘public static final’, and methods are ‘pu…
* [UnnecessaryReturn](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#unnecessaryreturn): Avoid the use of unnecessary return statements.
* [UseDiamondOperator](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#usediamondoperator): Use the diamond operator to let the type be inferred automatically. With the Diamond operator it …
* [UselessParentheses](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#uselessparentheses): Useless parentheses should be removed.
* [UselessQualifiedThis](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#uselessqualifiedthis): Reports qualified this usages in the same class.
* [UseShortArrayInitializer](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#useshortarrayinitializer): When declaring and initializing array fields or variables, it is not necessary to explicitly crea…
* [UseUnderscoresInNumericLiterals](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#useunderscoresinnumericliterals): Since Java 1.7, numeric literals can use underscores to separate digits. This rule enforces that …
* [VariableNamingConventions](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#variablenamingconventions): Deprecated A variable naming conventions rule - customize this to your liking. Currently, it checks for fin…
* [WhileLoopsMustUseBraces](https://pmd.github.io/latest/pmd_rules_java_codestyle.html#whileloopsmustusebraces): Deprecated Avoid using ‘while’ statements without using braces to surround the code block. If the code forma…

## **Design**

Rules that help you discover design issues.

* [AbstractClassWithoutAnyMethod](https://pmd.github.io/latest/pmd_rules_java_design.html#abstractclasswithoutanymethod): If an abstract class does not provides any methods, it may be acting as a simple data container t…
* [AvoidCatchingGenericException](https://pmd.github.io/latest/pmd_rules_java_design.html#avoidcatchinggenericexception): Avoid catching generic exceptions such as NullPointerException, RuntimeException, Exception in tr…
* [AvoidDeeplyNestedIfStmts](https://pmd.github.io/latest/pmd_rules_java_design.html#avoiddeeplynestedifstmts): Avoid creating deeply nested if-then statements since they are harder to read and error-prone to …
* [AvoidRethrowingException](https://pmd.github.io/latest/pmd_rules_java_design.html#avoidrethrowingexception): Catch blocks that merely rethrow a caught exception only add to code size and runtime complexity.
* [AvoidThrowingNewInstanceOfSameException](https://pmd.github.io/latest/pmd_rules_java_design.html#avoidthrowingnewinstanceofsameexception): Catch blocks that merely rethrow a caught exception wrapped inside a new instance of the same typ…
* [AvoidThrowingNullPointerException](https://pmd.github.io/latest/pmd_rules_java_design.html#avoidthrowingnullpointerexception): Avoid throwing NullPointerExceptions manually. These are confusing because most people will assum…
* [AvoidThrowingRawExceptionTypes](https://pmd.github.io/latest/pmd_rules_java_design.html#avoidthrowingrawexceptiontypes): Avoid throwing certain exception types. Rather than throw a raw RuntimeException, Throwable, Exce…
* [AvoidUncheckedExceptionsInSignatures](https://pmd.github.io/latest/pmd_rules_java_design.html#avoiduncheckedexceptionsinsignatures): A method or constructor should not explicitly declare unchecked exceptions in its ‘throws’ clause…
* [ClassWithOnlyPrivateConstructorsShouldBeFinal](https://pmd.github.io/latest/pmd_rules_java_design.html#classwithonlyprivateconstructorsshouldbefinal): A class with only private constructors should be final, unless the private constructor is invoked…
* [CollapsibleIfStatements](https://pmd.github.io/latest/pmd_rules_java_design.html#collapsibleifstatements): Sometimes two consecutive ‘if’ statements can be consolidated by separating their conditions with…
* [CouplingBetweenObjects](https://pmd.github.io/latest/pmd_rules_java_design.html#couplingbetweenobjects): This rule counts the number of unique attributes, local variables, and return types within an obj…
* [CyclomaticComplexity](https://pmd.github.io/latest/pmd_rules_java_design.html#cyclomaticcomplexity): The complexity of methods directly affects maintenance costs and readability. Concentrating too m…
* [DataClass](https://pmd.github.io/latest/pmd_rules_java_design.html#dataclass): Data Classes are simple data holders, which reveal most of their state, and without complex funct…
* [DoNotExtendJavaLangError](https://pmd.github.io/latest/pmd_rules_java_design.html#donotextendjavalangerror): Errors are system exceptions. Do not extend them.
* [ExceptionAsFlowControl](https://pmd.github.io/latest/pmd_rules_java_design.html#exceptionasflowcontrol): Using Exceptions as form of flow control is not recommended as they obscure true exceptions when …
* [ExcessiveClassLength](https://pmd.github.io/latest/pmd_rules_java_design.html#excessiveclasslength): Excessive class file lengths are usually indications that the class may be burdened with excessiv…
* [ExcessiveImports](https://pmd.github.io/latest/pmd_rules_java_design.html#excessiveimports): A high number of imports can indicate a high degree of coupling within an object. This rule count…
* [ExcessiveMethodLength](https://pmd.github.io/latest/pmd_rules_java_design.html#excessivemethodlength): When methods are excessively long this usually indicates that the method is doing more than its n…
* [ExcessiveParameterList](https://pmd.github.io/latest/pmd_rules_java_design.html#excessiveparameterlist): Methods with numerous parameters are a challenge to maintain, especially if most of them share th…
* [ExcessivePublicCount](https://pmd.github.io/latest/pmd_rules_java_design.html#excessivepubliccount): Classes with large numbers of public methods and attributes require disproportionate testing effo…
* [FinalFieldCouldBeStatic](https://pmd.github.io/latest/pmd_rules_java_design.html#finalfieldcouldbestatic): If a final field is assigned to a compile-time constant, it could be made static, thus saving ove…
* [GodClass](https://pmd.github.io/latest/pmd_rules_java_design.html#godclass): The God Class rule detects the God Class design flaw using metrics. God classes do too many thing…
* [ImmutableField](https://pmd.github.io/latest/pmd_rules_java_design.html#immutablefield): Identifies private fields whose values never change once object initialization ends either in the…
* [LawOfDemeter](https://pmd.github.io/latest/pmd_rules_java_design.html#lawofdemeter): The Law of Demeter is a simple rule, that says "only talk to friends". It helps to reduce couplin…
* [LogicInversion](https://pmd.github.io/latest/pmd_rules_java_design.html#logicinversion): Use opposite operator instead of negating the whole expression with a logic complement operator.
* [LoosePackageCoupling](https://pmd.github.io/latest/pmd_rules_java_design.html#loosepackagecoupling): Avoid using classes from the configured package hierarchy outside of the package hierarchy, excep…
* [ModifiedCyclomaticComplexity](https://pmd.github.io/latest/pmd_rules_java_design.html#modifiedcyclomaticcomplexity): Deprecated Complexity directly affects maintenance costs is determined by the number of decision points in a…
* [NcssConstructorCount](https://pmd.github.io/latest/pmd_rules_java_design.html#ncssconstructorcount): Deprecated This rule uses the NCSS (Non-Commenting Source Statements) algorithm to determine the number of l…
* [NcssCount](https://pmd.github.io/latest/pmd_rules_java_design.html#ncsscount): This rule uses the NCSS (Non-Commenting Source Statements) metric to determine the number of line…
* [NcssMethodCount](https://pmd.github.io/latest/pmd_rules_java_design.html#ncssmethodcount): Deprecated This rule uses the NCSS (Non-Commenting Source Statements) algorithm to determine the number of l…
* [NcssTypeCount](https://pmd.github.io/latest/pmd_rules_java_design.html#ncsstypecount): Deprecated This rule uses the NCSS (Non-Commenting Source Statements) algorithm to determine the number of l…
* [NPathComplexity](https://pmd.github.io/latest/pmd_rules_java_design.html#npathcomplexity): The NPath complexity of a method is the number of acyclic execution paths through that method. Wh…
* [SignatureDeclareThrowsException](https://pmd.github.io/latest/pmd_rules_java_design.html#signaturedeclarethrowsexception): A method/constructor shouldn’t explicitly throw the generic java.lang.Exception, since it is uncl…
* [SimplifiedTernary](https://pmd.github.io/latest/pmd_rules_java_design.html#simplifiedternary): Look for ternary operators with the form ‘condition ? literalBoolean : foo’ or ‘condition ? foo :…
* [SimplifyBooleanAssertion](https://pmd.github.io/latest/pmd_rules_java_design.html#simplifybooleanassertion): Avoid negation in an assertTrue or assertFalse test. For example, rephrase: assertTrue(!expr)…
* [SimplifyBooleanExpressions](https://pmd.github.io/latest/pmd_rules_java_design.html#simplifybooleanexpressions): Avoid unnecessary comparisons in boolean expressions, they serve no purpose and impacts readability.
* [SimplifyBooleanReturns](https://pmd.github.io/latest/pmd_rules_java_design.html#simplifybooleanreturns): Avoid unnecessary if-then-else statements when returning a boolean. The result of the conditional…
* [SimplifyConditional](https://pmd.github.io/latest/pmd_rules_java_design.html#simplifyconditional): No need to check for null before an instanceof; the instanceof keyword returns false when given a…
* [SingularField](https://pmd.github.io/latest/pmd_rules_java_design.html#singularfield): Fields whose scopes are limited to just single methods do not rely on the containing object to pr…
* [StdCyclomaticComplexity](https://pmd.github.io/latest/pmd_rules_java_design.html#stdcyclomaticcomplexity): Deprecated Complexity directly affects maintenance costs is determined by the number of decision points in a…
* [SwitchDensity](https://pmd.github.io/latest/pmd_rules_java_design.html#switchdensity): A high ratio of statements to labels in a switch statement implies that the switch statement is o…
* [TooManyFields](https://pmd.github.io/latest/pmd_rules_java_design.html#toomanyfields): Classes that have too many fields can become unwieldy and could be redesigned to have fewer field…
* [TooManyMethods](https://pmd.github.io/latest/pmd_rules_java_design.html#toomanymethods): A class with too many methods is probably a good suspect for refactoring, in order to reduce its …
* [UselessOverridingMethod](https://pmd.github.io/latest/pmd_rules_java_design.html#uselessoverridingmethod): The overriding method merely calls the same method defined in a superclass.
* [UseObjectForClearerAPI](https://pmd.github.io/latest/pmd_rules_java_design.html#useobjectforclearerapi): When you write a public method, you should be thinking in terms of an API. If your method is publ…
* [UseUtilityClass](https://pmd.github.io/latest/pmd_rules_java_design.html#useutilityclass): For classes that only have static methods, consider making them utility classes. Note that this d…

## **Documentation**

Rules that are related to code documentation.

* [CommentContent](https://pmd.github.io/latest/pmd_rules_java_documentation.html#commentcontent): A rule for the politically correct… we don’t want to offend anyone.
* [CommentRequired](https://pmd.github.io/latest/pmd_rules_java_documentation.html#commentrequired): Denotes whether javadoc (formal) comments are required (or unwanted) for specific language elements.
* [CommentSize](https://pmd.github.io/latest/pmd_rules_java_documentation.html#commentsize): Determines whether the dimensions of non-header comments found are within the specified limits.
* [UncommentedEmptyConstructor](https://pmd.github.io/latest/pmd_rules_java_documentation.html#uncommentedemptyconstructor): Uncommented Empty Constructor finds instances where a constructor does not contain statements, bu…
* [UncommentedEmptyMethodBody](https://pmd.github.io/latest/pmd_rules_java_documentation.html#uncommentedemptymethodbody): Uncommented Empty Method Body finds instances where a method body does not contain statements, bu…

## **Error Prone**

Rules to detect constructs that are either broken, extremely confusing or prone to runtime errors.

* [AssignmentInOperand](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#assignmentinoperand): Avoid assignments in operands; this can make code more complicated and harder to read.
* [AssignmentToNonFinalStatic](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#assignmenttononfinalstatic): Identifies a possible unsafe usage of a static field.
* [AvoidAccessibilityAlteration](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidaccessibilityalteration): Methods such as getDeclaredConstructors(), getDeclaredConstructor(Class[]) and setAccessible(), a…
* [AvoidAssertAsIdentifier](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidassertasidentifier): Use of the term ‘assert’ will conflict with newer versions of Java since it is a reserved word.
* [AvoidBranchingStatementAsLastInLoop](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidbranchingstatementaslastinloop): Using a branching statement as the last part of a loop may be a bug, and/or is confusing. Ensure …
* [AvoidCallingFinalize](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidcallingfinalize): The method Object.finalize() is called by the garbage collector on an object when garbage collect…
* [AvoidCatchingNPE](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidcatchingnpe): Code should never throw NullPointerExceptions under normal circumstances. A catch block may hide…
* [AvoidCatchingThrowable](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidcatchingthrowable): Catching Throwable errors is not recommended since its scope is very broad. It includes runtime i…
* [AvoidDecimalLiteralsInBigDecimalConstructor](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoiddecimalliteralsinbigdecimalconstructor): One might assume that the result of "new BigDecimal(0.1)" is exactly equal to 0.1, but it is actu…
* [AvoidDuplicateLiterals](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidduplicateliterals): Code containing duplicate String literals can usually be improved by declaring the String as a co…
* [AvoidEnumAsIdentifier](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidenumasidentifier): Use of the term ‘enum’ will conflict with newer versions of Java since it is a reserved word.
* [AvoidFieldNameMatchingMethodName](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidfieldnamematchingmethodname): It can be confusing to have a field name with the same name as a method. While this is permitted,…
* [AvoidFieldNameMatchingTypeName](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidfieldnamematchingtypename): It is somewhat confusing to have a field name matching the declaring class name. This probably me…
* [AvoidInstanceofChecksInCatchClause](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidinstanceofchecksincatchclause): Each caught exception type should be handled in its own catch clause.
* [AvoidLiteralsInIfCondition](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidliteralsinifcondition): Avoid using hard-coded literals in conditional statements. By declaring them as static variables …
* [AvoidLosingExceptionInformation](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidlosingexceptioninformation): Statements in a catch block that invoke accessors on the exception without using the information …
* [AvoidMultipleUnaryOperators](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidmultipleunaryoperators): The use of multiple unary operators may be problematic, and/or confusing. Ensure that the intende…
* [AvoidUsingOctalValues](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#avoidusingoctalvalues): Integer literals should not start with zero since this denotes that the rest of literal will be i…
* [BadComparison](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#badcomparison): Avoid equality comparisons with Double.NaN. Due to the implicit lack of representation precision …
* [BeanMembersShouldSerialize](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#beanmembersshouldserialize): If a class is a bean, or is referenced by a bean directly or indirectly it needs to be serializab…
* [BrokenNullCheck](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#brokennullcheck): The null check is broken since it will throw a NullPointerException itself. It is likely that you…
* [CallSuperFirst](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#callsuperfirst): Super should be called at the start of the method
* [CallSuperLast](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#callsuperlast): Super should be called at the end of the method
* [CheckSkipResult](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#checkskipresult): The skip() method may skip a smaller number of bytes than requested. Check the returned value to …
* [ClassCastExceptionWithToArray](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#classcastexceptionwithtoarray): When deriving an array of a specific class from your Collection, one should provide an array of t…
* [CloneMethodMustBePublic](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#clonemethodmustbepublic): The java Manual says "By convention, classes that implement this interface should override Object…
* [CloneMethodMustImplementCloneable](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#clonemethodmustimplementcloneable): The method clone() should only be implemented if the class implements the Cloneable interface wit…
* [CloneMethodReturnTypeMustMatchClassName](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#clonemethodreturntypemustmatchclassname): If a class implements cloneable the return type of the method clone() must be the class name. Tha…
* [CloneThrowsCloneNotSupportedException](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#clonethrowsclonenotsupportedexception): The method clone() should throw a CloneNotSupportedException.
* [CloseResource](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#closeresource): Ensure that resources (like ‘java.sql.Connection’, ‘java.sql.Statement’, and ‘java.sql.ResultSet’…
* [CompareObjectsWithEquals](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#compareobjectswithequals): Use equals() to compare object references; avoid comparing them with ==.
* [ConstructorCallsOverridableMethod](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#constructorcallsoverridablemethod): Calling overridable methods during construction poses a risk of invoking methods on an incomplete…
* [DataflowAnomalyAnalysis](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#dataflowanomalyanalysis): Deprecated The dataflow analysis tracks local definitions, undefinitions and references to variables on diff…
* [DetachedTestCase](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#detachedtestcase): The method appears to be a test case since it has public or default visibility, non-static access…
* [DoNotCallGarbageCollectionExplicitly](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#donotcallgarbagecollectionexplicitly): Calls to System.gc(), Runtime.getRuntime().gc(), and System.runFinalization() are not advised. Co…
* [DoNotExtendJavaLangThrowable](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#donotextendjavalangthrowable): Extend Exception or RuntimeException instead of Throwable.
* [DoNotHardCodeSDCard](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#donothardcodesdcard): Use Environment.getExternalStorageDirectory() instead of "/sdcard"
* [DoNotTerminateVM](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#donotterminatevm): Web applications should not call ‘System.exit()’, since only the web container or the application…
* [DoNotThrowExceptionInFinally](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#donotthrowexceptioninfinally): Throwing exceptions within a ‘finally’ block is confusing since they may mask other exceptions or…
* [DontImportSun](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#dontimportsun): Avoid importing anything from the ‘sun.’ packages. These packages are not portable and are likel…
* [DontUseFloatTypeForLoopIndices](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#dontusefloattypeforloopindices): Don’t use floating point for loop indices. If you must use floating point, use double unless you’…
* [EmptyCatchBlock](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptycatchblock): Empty Catch Block finds instances where an exception is caught, but nothing is done. In most circ…
* [EmptyFinalizer](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptyfinalizer): Empty finalize methods serve no purpose and should be removed. Note that Oracle has declared Obje…
* [EmptyFinallyBlock](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptyfinallyblock): Empty finally blocks serve no purpose and should be removed.
* [EmptyIfStmt](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptyifstmt): Empty If Statement finds instances where a condition is checked but nothing is done about it.
* [EmptyInitializer](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptyinitializer): Empty initializers serve no purpose and should be removed.
* [EmptyStatementBlock](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptystatementblock): Empty block statements serve no purpose and should be removed.
* [EmptyStatementNotInLoop](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptystatementnotinloop): An empty statement (or a semicolon by itself) that is not used as the sole body of a ‘for’ or ‘wh…
* [EmptySwitchStatements](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptyswitchstatements): Empty switch statements serve no purpose and should be removed.
* [EmptySynchronizedBlock](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptysynchronizedblock): Empty synchronized blocks serve no purpose and should be removed.
* [EmptyTryBlock](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptytryblock): Avoid empty try blocks - what’s the point?
* [EmptyWhileStmt](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#emptywhilestmt): Empty While Statement finds all instances where a while statement does nothing. If it is a timing…
* [EqualsNull](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#equalsnull): Tests for null should not use the equals() method. The ‘==’ operator should be used instead.
* [FinalizeDoesNotCallSuperFinalize](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#finalizedoesnotcallsuperfinalize): If the finalize() is implemented, its last action should be to call super.finalize. Note that Ora…
* [FinalizeOnlyCallsSuperFinalize](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#finalizeonlycallssuperfinalize): If the finalize() is implemented, it should do something besides just calling super.finalize(). N…
* [FinalizeOverloaded](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#finalizeoverloaded): Methods named finalize() should not have parameters. It is confusing and most likely an attempt …
* [FinalizeShouldBeProtected](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#finalizeshouldbeprotected): When overriding the finalize(), the new method should be set as protected. If made public, other…
* [IdempotentOperations](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#idempotentoperations): Avoid idempotent operations - they have no effect.
* [ImportFromSamePackage](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#importfromsamepackage): There is no need to import a type that lives in the same package.
* [InstantiationToGetClass](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#instantiationtogetclass): Avoid instantiating an object just to call getClass() on it; use the .class public member instead.
* [InvalidLogMessageFormat](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#invalidlogmessageformat): Check for messages in slf4j and log4j2 (since 6.19.0) loggers with non matching number of argumen…
* [JumbledIncrementer](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#jumbledincrementer): Avoid jumbled loop incrementers - its usually a mistake, and is confusing even if intentional.
* [JUnitSpelling](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#junitspelling): In JUnit 3, the setUp method is used to set up all data entities required in running tests. …
* [JUnitStaticSuite](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#junitstaticsuite): The suite() method in a JUnit test needs to be both public and static.
* [LoggerIsNotStaticFinal](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#loggerisnotstaticfinal): Deprecated In most cases, the Logger reference can be declared as static and final. This rule is deprecated …
* [MethodWithSameNameAsEnclosingClass](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#methodwithsamenameasenclosingclass): Non-constructor methods should not have the same name as the enclosing class.
* [MisplacedNullCheck](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#misplacednullcheck): The null check here is misplaced. If the variable is null a NullPointerException will be thrown. …
* [MissingBreakInSwitch](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#missingbreakinswitch): Switch statements without break or return statements for each case option may indicate problemati…
* [MissingSerialVersionUID](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#missingserialversionuid): Serializable classes should provide a serialVersionUID field. The serialVersionUID field is also …
* [MissingStaticMethodInNonInstantiatableClass](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#missingstaticmethodinnoninstantiatableclass): A class that has private constructors and does not have any static methods or fields cannot be us…
* [MoreThanOneLogger](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#morethanonelogger): Normally only one logger is used in each class. This rule supports slf4j, log4j, Java Util Loggin…
* [NonCaseLabelInSwitchStatement](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#noncaselabelinswitchstatement): A non-case label (e.g. a named break/continue label) was present in a switch statement. This lega…
* [NonStaticInitializer](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#nonstaticinitializer): A non-static initializer block will be called any time a constructor is invoked (just prior to in…
* [NullAssignment](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#nullassignment): Assigning a "null" to a variable (outside of its declaration) is usually bad form. Sometimes, th…
* [OverrideBothEqualsAndHashcode](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#overridebothequalsandhashcode): Override both public boolean Object.equals(Object other), and public int Object.hashCode(), or ov…
* [ProperCloneImplementation](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#propercloneimplementation): Object clone() should be implemented with super.clone().
* [ProperLogger](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#properlogger): A logger should normally be defined private static final and be associated with the correct class…
* [ReturnEmptyArrayRatherThanNull](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#returnemptyarrayratherthannull): For any method that returns an array, it is a better to return an empty array rather than a null …
* [ReturnFromFinallyBlock](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#returnfromfinallyblock): Avoid returning from a finally block, this can discard exceptions.
* [SimpleDateFormatNeedsLocale](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#simpledateformatneedslocale): Be sure to specify a Locale when creating SimpleDateFormat instances to ensure that locale-approp…
* [SingleMethodSingleton](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#singlemethodsingleton): Some classes contain overloaded getInstance. The problem with overloaded getInstance methods is t…
* [SingletonClassReturningNewInstance](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#singletonclassreturningnewinstance): Some classes contain overloaded getInstance. The problem with overloaded getInstance methods is t…
* [StaticEJBFieldShouldBeFinal](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#staticejbfieldshouldbefinal): According to the J2EE specification, an EJB should not have any static fields with write access. …
* [StringBufferInstantiationWithChar](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#stringbufferinstantiationwithchar): Individual character values provided as initialization arguments will be converted into integers….
* [SuspiciousEqualsMethodName](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#suspiciousequalsmethodname): The method name and parameter number are suspiciously close to ‘Object.equals’, which can denote …
* [SuspiciousHashcodeMethodName](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#suspicioushashcodemethodname): The method name and return type are suspiciously close to hashCode(), which may denote an intenti…
* [SuspiciousOctalEscape](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#suspiciousoctalescape): A suspicious octal escape sequence was found inside a String literal. The Java language specifica…
* [TestClassWithoutTestCases](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#testclasswithouttestcases): Test classes end with the suffix Test. Having a non-test class with that name is not a good pract…
* [UnconditionalIfStatement](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#unconditionalifstatement): Do not use "if" statements whose conditionals are always true or always false.
* [UnnecessaryBooleanAssertion](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#unnecessarybooleanassertion): A JUnit test assertion with a boolean literal is unnecessary since it always will evaluate to the…
* [UnnecessaryCaseChange](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#unnecessarycasechange): Using equalsIgnoreCase() is faster than using toUpperCase/toLowerCase().equals()
* [UnnecessaryConversionTemporary](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#unnecessaryconversiontemporary): Avoid the use temporary objects when converting primitives to Strings. Use the static conversion …
* [UnusedNullCheckInEquals](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#unusednullcheckinequals): After checking an object reference for null, you should invoke equals() on that object rather tha…
* [UseCorrectExceptionLogging](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#usecorrectexceptionlogging): To make sure the full stacktrace is printed out, use the logging statement with two arguments: a …
* [UseEqualsToCompareStrings](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#useequalstocomparestrings): Using ‘==’ or ‘!=’ to compare strings only works if intern version is used on both sides. Use the…
* [UselessOperationOnImmutable](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#uselessoperationonimmutable): An operation on an Immutable object (String, BigDecimal or BigInteger) won’t change the object it…
* [UseLocaleWithCaseConversions](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#uselocalewithcaseconversions): When doing ‘String::toLowerCase()/toUpperCase()’ conversions, use an explicit locale argument to …
* [UseProperClassLoader](https://pmd.github.io/latest/pmd_rules_java_errorprone.html#useproperclassloader): In J2EE, the getClassLoader() method might not work as expected. Use Thread.currentThread().getCo…

## **Multithreading**

Rules that flag issues when dealing with multiple threads of execution.

* [AvoidSynchronizedAtMethodLevel](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#avoidsynchronizedatmethodlevel): Method-level synchronization can cause problems when new code is added to the method. Block-level…
* [AvoidThreadGroup](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#avoidthreadgroup): Avoid using java.lang.ThreadGroup; although it is intended to be used in a threaded environment i…
* [AvoidUsingVolatile](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#avoidusingvolatile): Use of the keyword ‘volatile’ is generally used to fine tune a Java application, and therefore, r…
* [DoNotUseThreads](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#donotusethreads): The J2EE specification explicitly forbids the use of threads. Threads are resources, that should …
* [DontCallThreadRun](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#dontcallthreadrun): Explicitly calling Thread.run() method will execute in the caller’s thread of control. Instead, …
* [DoubleCheckedLocking](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#doublecheckedlocking): Partially created objects can be returned by the Double Checked Locking pattern when used in Java…
* [NonThreadSafeSingleton](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#nonthreadsafesingleton): Non-thread safe singletons can result in bad state changes. Eliminate static singletons if possib…
* [UnsynchronizedStaticDateFormatter](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#unsynchronizedstaticdateformatter): Deprecated SimpleDateFormat instances are not synchronized. Sun recommends using separate format instances f…
* [UnsynchronizedStaticFormatter](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#unsynchronizedstaticformatter): Instances of ‘java.text.Format’ are generally not synchronized. Sun recommends using separate for…
* [UseConcurrentHashMap](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#useconcurrenthashmap): Since Java5 brought a new implementation of the Map designed for multi-threaded access, you can p…
* [UseNotifyAllInsteadOfNotify](https://pmd.github.io/latest/pmd_rules_java_multithreading.html#usenotifyallinsteadofnotify): Thread.notify() awakens a thread monitoring the object. If more than one thread is monitoring, th…

## **Performance**

Rules that flag suboptimal code.

* [AddEmptyString](https://pmd.github.io/latest/pmd_rules_java_performance.html#addemptystring): The conversion of literals to strings by concatenating them with empty strings is inefficient. It…
* [AppendCharacterWithChar](https://pmd.github.io/latest/pmd_rules_java_performance.html#appendcharacterwithchar): Avoid concatenating characters as strings in StringBuffer/StringBuilder.append methods.
* [AvoidArrayLoops](https://pmd.github.io/latest/pmd_rules_java_performance.html#avoidarrayloops): Instead of manually copying data between two arrays, use the efficient Arrays.copyOf or System.ar…
* [AvoidCalendarDateCreation](https://pmd.github.io/latest/pmd_rules_java_performance.html#avoidcalendardatecreation): Problem: A Calendar is a heavyweight object and expensive to create. Solution: Use ‘new Date()’, …
* [AvoidFileStream](https://pmd.github.io/latest/pmd_rules_java_performance.html#avoidfilestream): The FileInputStream and FileOutputStream classes contains a finalizer method which will cause gar…
* [AvoidInstantiatingObjectsInLoops](https://pmd.github.io/latest/pmd_rules_java_performance.html#avoidinstantiatingobjectsinloops): New objects created within loops should be checked to see if they can created outside them and re…
* [AvoidUsingShortType](https://pmd.github.io/latest/pmd_rules_java_performance.html#avoidusingshorttype): Deprecated Note: this rule is deprecated, as its rationale does not hold. Java uses the ‘short’ type to redu…
* [BigIntegerInstantiation](https://pmd.github.io/latest/pmd_rules_java_performance.html#bigintegerinstantiation): Don’t create instances of already existing BigInteger (BigInteger.ZERO, BigInteger.ONE) and for J…
* [BooleanInstantiation](https://pmd.github.io/latest/pmd_rules_java_performance.html#booleaninstantiation): Avoid instantiating Boolean objects; you can reference Boolean.TRUE, Boolean.FALSE, or call Boole…
* [ByteInstantiation](https://pmd.github.io/latest/pmd_rules_java_performance.html#byteinstantiation): Calling new Byte() causes memory allocation that can be avoided by the static Byte.valueOf(). It …
* [ConsecutiveAppendsShouldReuse](https://pmd.github.io/latest/pmd_rules_java_performance.html#consecutiveappendsshouldreuse): Consecutive calls to StringBuffer/StringBuilder .append should be chained, reusing the target obj…
* [ConsecutiveLiteralAppends](https://pmd.github.io/latest/pmd_rules_java_performance.html#consecutiveliteralappends): Consecutively calling StringBuffer/StringBuilder.append(…) with literals should be avoided. Sin…
* [InefficientEmptyStringCheck](https://pmd.github.io/latest/pmd_rules_java_performance.html#inefficientemptystringcheck): String.trim().length() == 0 (or String.trim().isEmpty() for the same reason) is an inefficient wa…
* [InefficientStringBuffering](https://pmd.github.io/latest/pmd_rules_java_performance.html#inefficientstringbuffering): Avoid concatenating non-literals in a StringBuffer constructor or append() since intermediate buf…
* [InsufficientStringBufferDeclaration](https://pmd.github.io/latest/pmd_rules_java_performance.html#insufficientstringbufferdeclaration): Failing to pre-size a StringBuffer or StringBuilder properly could cause it to re-size many times…
* [IntegerInstantiation](https://pmd.github.io/latest/pmd_rules_java_performance.html#integerinstantiation): Calling new Integer() causes memory allocation that can be avoided by the static Integer.valueOf(…
* [LongInstantiation](https://pmd.github.io/latest/pmd_rules_java_performance.html#longinstantiation): Calling new Long() causes memory allocation that can be avoided by the static Long.valueOf(). It …
* [OptimizableToArrayCall](https://pmd.github.io/latest/pmd_rules_java_performance.html#optimizabletoarraycall): Calls to a collection’s ‘toArray(E[])’ method should specify a target array of zero size. This al…
* [RedundantFieldInitializer](https://pmd.github.io/latest/pmd_rules_java_performance.html#redundantfieldinitializer): Java will initialize fields with known default values so any explicit initialization of those sam…
* [ShortInstantiation](https://pmd.github.io/latest/pmd_rules_java_performance.html#shortinstantiation): Calling new Short() causes memory allocation that can be avoided by the static Short.valueOf(). I…
* [SimplifyStartsWith](https://pmd.github.io/latest/pmd_rules_java_performance.html#simplifystartswith): Deprecated Note: this rule is deprecated for removal, as the optimization is insignificant. Calls to ‘string…
* [StringInstantiation](https://pmd.github.io/latest/pmd_rules_java_performance.html#stringinstantiation): Avoid instantiating String objects; this is usually unnecessary since they are immutable and can …
* [StringToString](https://pmd.github.io/latest/pmd_rules_java_performance.html#stringtostring): Avoid calling toString() on objects already known to be string instances; this is unnecessary.
* [TooFewBranchesForASwitchStatement](https://pmd.github.io/latest/pmd_rules_java_performance.html#toofewbranchesforaswitchstatement): Switch statements are intended to be used to support complex branching behaviour. Using a switch …
* [UnnecessaryWrapperObjectCreation](https://pmd.github.io/latest/pmd_rules_java_performance.html#unnecessarywrapperobjectcreation): Most wrapper classes provide static conversion methods that avoid the need to create intermediate…
* [UseArrayListInsteadOfVector](https://pmd.github.io/latest/pmd_rules_java_performance.html#usearraylistinsteadofvector): ArrayList is a much better Collection implementation than Vector if thread-safe operation is not …
* [UseArraysAsList](https://pmd.github.io/latest/pmd_rules_java_performance.html#usearraysaslist): The java.util.Arrays class has a "asList" method that should be used when you want to create a ne…
* [UseIndexOfChar](https://pmd.github.io/latest/pmd_rules_java_performance.html#useindexofchar): Use String.indexOf(char) when checking for the index of a single character; it executes faster.
* [UseIOStreamsWithApacheCommonsFileItem](https://pmd.github.io/latest/pmd_rules_java_performance.html#useiostreamswithapachecommonsfileitem): Problem: Use of [FileItem.get()](https://commons.apache.org/proper/commons-fileupload/apidocs/org…
* [UselessStringValueOf](https://pmd.github.io/latest/pmd_rules_java_performance.html#uselessstringvalueof): No need to call String.valueOf to append to a string; just use the valueOf() argument directly.
* [UseStringBufferForStringAppends](https://pmd.github.io/latest/pmd_rules_java_performance.html#usestringbufferforstringappends): The use of the ‘+=’ operator for appending strings causes the JVM to create and use an internal S…
* [UseStringBufferLength](https://pmd.github.io/latest/pmd_rules_java_performance.html#usestringbufferlength): Use StringBuffer.length() to determine StringBuffer length rather than using StringBuffer.toStrin…

## **Security**

Rules that flag potential security flaws.

* [HardCodedCryptoKey](https://pmd.github.io/latest/pmd_rules_java_security.html#hardcodedcryptokey): Do not use hard coded values for cryptographic operations. Please store keys outside of source code.
* [InsecureCryptoIv](https://pmd.github.io/latest/pmd_rules_java_security.html#insecurecryptoiv): Do not use hard coded initialization vector in cryptographic operations. Please use a randomly ge…