Best Practices

Rules which enforce generally accepted best practices.

ApexAssertionsShouldIncludeMessage

**Since:** PMD 6.13.0

**Priority:** Medium (3)

The second parameter of System.assert/third parameter of System.assertEquals/System.assertNotEquals is a message. Having a second/third parameter provides more information and makes it easier to debug the test failure and improves the readability of test output.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.bestpractices.ApexAssertionsShouldIncludeMessageRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/bestpractices/ApexAssertionsShouldIncludeMessageRule.java)

**Example(s):**

@isTest

**public** **class** **Foo** **{**

@isTest

**static** **void** **methodATest()** **{**

**System.**assertNotEquals**(**'123'**,** o**.**StageName**);** *// not good*

**System.**assertEquals**(**'123'**,** o**.**StageName**,** '**Opportunity** stageName is wrong**.**'**);** *// good*

**System.**assert**(**o**.**isClosed**);** *// not good*

**System.**assert**(**o**.**isClosed**,** '**Opportunity** is not closed**.**'**);** *// good*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Bug Risk | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/bestpractices.xml/ApexAssertionsShouldIncludeMessage" />

ApexUnitTestClassShouldHaveAsserts

**Since:** PMD 5.5.1

**Priority:** Medium (3)

Apex unit tests should include at least one assertion. This makes the tests more robust, and using assert with messages provide the developer a clearer idea of what the test does. Custom assert method invocation patterns can be specified using the ‘additionalAssertMethodPattern’ property if required.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.bestpractices.ApexUnitTestClassShouldHaveAssertsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/bestpractices/ApexUnitTestClassShouldHaveAssertsRule.java)

**Example(s):**

@isTest

**public** **class** **Foo** **{**

**public** **static** testMethod **void** **testSomething()** **{**

**Account** a **=** **null;**

*// This is better than having a NullPointerException*

*// System.assertNotEquals(a, null, 'account not found');*

a**.**toString**();**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Bug Risk | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| additionalAssertMethodPattern |  | A regular expression for one or more custom test assertion method patterns. | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/bestpractices.xml/ApexUnitTestClassShouldHaveAsserts" />

**Use this rule and customize it:**

<rule ref="category/apex/bestpractices.xml/ApexUnitTestClassShouldHaveAsserts">

<properties>

<property name="additionalAssertMethodPattern" value="" />

</properties>

</rule>

ApexUnitTestMethodShouldHaveIsTestAnnotation

**Since:** PMD 6.13.0

**Priority:** Medium (3)

Apex test methods should have @isTest annotation instead of the testMethod keyword, as testMethod is deprecated. Salesforce advices to use [@isTest](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_annotation_isTest.htm) annotation for test classes and methods.

**This rule is defined by the following XPath expression:**

**//**Method[ModifierNode[@DeprecatedTestMethod = **true**()]]

**Example(s):**

@isTest

**private** **class** **ATest** **{**

@isTest

**static** **void** **methodATest()** **{**

**}**

**static** **void** **methodBTest()** **{**

**}**

@isTest **static** **void** **methodCTest()** **{**

**System.**assert**(**1**==**2**);**

**}**

**static** testmethod **void** **methodCTest()** **{**

**System.**debug**(**'I am a debug statement'**);**

**}**

**private** **void** **fetchData()** **{**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/bestpractices.xml/ApexUnitTestMethodShouldHaveIsTestAnnotation" />

ApexUnitTestShouldNotUseSeeAllDataTrue

**Since:** PMD 5.5.1

**Priority:** Medium (3)

Apex unit tests should not use @isTest(seeAllData=true) because it opens up the existing database data for unexpected modification by tests.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.bestpractices.ApexUnitTestShouldNotUseSeeAllDataTrueRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/bestpractices/ApexUnitTestShouldNotUseSeeAllDataTrueRule.java)

**Example(s):**

@isTest**(**seeAllData **=** **true)**

**public** **class** **Foo** **{**

**public** **static** testMethod **void** **testSomething()** **{**

**Account** a **=** **null;**

*// This is better than having a NullPointerException*

*// System.assertNotEquals(a, null, 'account not found');*

a**.**toString**();**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Bug Risk | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/bestpractices.xml/ApexUnitTestShouldNotUseSeeAllDataTrue" />

AvoidGlobalModifier

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Global classes should be avoided (especially in managed packages) as they can never be deleted or changed in signature. Always check twice if something needs to be global. Many interfaces (e.g. Batch) required global modifiers in the past but don’t require this anymore. Don’t lock yourself in.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.bestpractices.AvoidGlobalModifierRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/bestpractices/AvoidGlobalModifierRule.java)

**Example(s):**

global **class** **Unchangeable** **{**

global **UndeletableType** **unchangable(UndeletableType** param**)** **{**

*// ...*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/bestpractices.xml/AvoidGlobalModifier" />

AvoidLogicInTrigger

**Since:** PMD 5.5.0

**Priority:** Medium (3)

As triggers do not allow methods like regular classes they are less flexible and suited to apply good encapsulation style. Therefore delegate the triggers work to a regular class (often called Trigger handler class).

See more here: <https://developer.salesforce.com/page/Trigger_Frameworks_and_Apex_Trigger_Best_Practices>

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.bestpractices.AvoidLogicInTriggerRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/bestpractices/AvoidLogicInTriggerRule.java)

**Example(s):**

trigger **Accounts** on **Account** **(**before insert**,** before update**,** before delete**,** after insert**,** after update**,** after delete**,** after undelete**)** **{**

**for(Account** acc **:** **Trigger.**new**)** **{**

**if(Trigger.**isInsert**)** **{**

*// ...*

**}**

*// ...*

**if(Trigger.**isDelete**)** **{**

*// ...*

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 200 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/bestpractices.xml/AvoidLogicInTrigger" />

DebugsShouldUseLoggingLevel

**Since:** PMD 6.18.0

**Priority:** Medium (3)

The first parameter of System.debug, when using the signature with two parameters, is a LoggingLevel enum.

Having the Logging Level specified provides a cleaner log, and improves readability of it.

**This rule is defined by the following XPath expression:**

**//**MethodCallExpression[**lower-case**(@FullMethodName)='system.debug'][**count**(\*)=2

**or** ($strictMode=**true**() **and** **count**(\*)=3 **and** **lower-case**(VariableExpression/@Image)='debug')]

**Example(s):**

@isTest

**public** **class** **Foo** **{**

@isTest

**static** **void** **bar()** **{**

**System.**debug**(**'**Hey** **this** code executed**.**'**);** *// not good*

**System.**debug**(LoggingLevel.**WARN**,** '**Hey,** something might be wrong**.**'**);** *// good*

**System.**debug**(LoggingLevel.**DEBUG**,** '**Hey,** something happened**.**'**);** *// not good when on strict mode*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| strictMode | false | If true, mark statements that use the DEBUG enum of LoggingLevel. | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/bestpractices.xml/DebugsShouldUseLoggingLevel" />

**Use this rule and customize it:**

<rule ref="category/apex/bestpractices.xml/DebugsShouldUseLoggingLevel">

<properties>

<property name="strictMode" value="false" />

</properties>

</rule>

UnusedLocalVariable

**Since:** PMD 6.23.0

**Priority:** Low (5)

Detects when a local variable is declared and/or assigned but not used.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.bestpractices.UnusedLocalVariableRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/bestpractices/UnusedLocalVariableRule.java)

**Example(s):**

**public** **Boolean** **bar(String** z**)** **{**

**String** x **=** 'some string'**;** *// not used*

**String** y **=** 'some other string'**;** *// used in the next line*

**return** z**.**equals**(**y**);**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

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

Code Style

Rules which enforce a specific coding style.

ClassNamingConventions

**Since:** PMD 5.5.0

**Priority:** High (1)

Configurable naming conventions for type declarations. This rule reports type declarations which do not match the regex that applies to their specific kind (e.g. enum or interface). Each regex can be configured through properties.

By default this rule uses the standard Apex naming convention (Pascal case).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.codestyle.ClassNamingConventionsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/codestyle/ClassNamingConventionsRule.java)

**Example(s):**

**public** **class** **FooClass** **{** **}** *// This is in pascal case, so it's ok*

**public** **class** **fooClass** **{** **}** *// This will be reported unless you change the regex*

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 5 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| testClassPattern | [A-Z][a-zA-Z0-9\_]\* | Regex which applies to test class names | no |
| abstractClassPattern | [A-Z][a-zA-Z0-9\_]\* | Regex which applies to abstract class names | no |
| classPattern | [A-Z][a-zA-Z0-9\_]\* | Regex which applies to class names | no |
| interfacePattern | [A-Z][a-zA-Z0-9\_]\* | Regex which applies to interface names | no |
| enumPattern | [A-Z][a-zA-Z0-9\_]\* | Regex which applies to enum names | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/ClassNamingConventions" />

**Use this rule and customize it:**

<rule ref="category/apex/codestyle.xml/ClassNamingConventions">

<properties>

<property name="testClassPattern" value="[A-Z][a-zA-Z0-9\_]\*" />

<property name="abstractClassPattern" value="[A-Z][a-zA-Z0-9\_]\*" />

<property name="classPattern" value="[A-Z][a-zA-Z0-9\_]\*" />

<property name="interfacePattern" value="[A-Z][a-zA-Z0-9\_]\*" />

<property name="enumPattern" value="[A-Z][a-zA-Z0-9\_]\*" />

</properties>

</rule>

FieldDeclarationsShouldBeAtStart

**Since:** PMD 6.23.0

**Priority:** Medium (3)

Field declarations should appear before method declarations within a class.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.codestyle.FieldDeclarationsShouldBeAtStartRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/codestyle/FieldDeclarationsShouldBeAtStartRule.java)

**Example(s):**

**class** **Foo** **{**

**public** **Integer** someField**;** *// good*

**public** **void** **someMethod()** **{**

**}**

**public** **Integer** anotherField**;** *// bad*

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/FieldDeclarationsShouldBeAtStart" />

FieldNamingConventions

**Since:** PMD 6.15.0

**Priority:** High (1)

Configurable naming conventions for field declarations. This rule reports variable declarations which do not match the regex that applies to their specific kind —e.g. constants (static final), static field, final field. Each regex can be configured through properties.

By default this rule uses the standard Apex naming convention (Camel case).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.codestyle.FieldNamingConventionsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/codestyle/FieldNamingConventionsRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**Integer** instanceField**;** *// This is in camel case, so it's ok*

**Integer** INSTANCE\_FIELD**;** *// This will be reported unless you change the regex*

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| enumConstantPattern | [A-Z][A-Z0-9\_]\* | Regex which applies to enum constant field names | no |
| constantPattern | [A-Z][A-Z0-9\_]\* | Regex which applies to constant field names | no |
| finalPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to final field names | no |
| staticPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to static field names | no |
| instancePattern | [a-z][a-zA-Z0-9]\* | Regex which applies to instance field names | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/FieldNamingConventions" />

**Use this rule and customize it:**

<rule ref="category/apex/codestyle.xml/FieldNamingConventions">

<properties>

<property name="enumConstantPattern" value="[A-Z][A-Z0-9\_]\*" />

<property name="constantPattern" value="[A-Z][A-Z0-9\_]\*" />

<property name="finalPattern" value="[a-z][a-zA-Z0-9]\*" />

<property name="staticPattern" value="[a-z][a-zA-Z0-9]\*" />

<property name="instancePattern" value="[a-z][a-zA-Z0-9]\*" />

</properties>

</rule>

ForLoopsMustUseBraces

**Since:** PMD 5.6.0

**Priority:** Medium (3)

Avoid using ‘for’ statements without using surrounding braces. If the code formatting or indentation is lost then it becomes difficult to separate the code being controlled from the rest.

**This rule is defined by the following XPath expression:**

**//**ForLoopStatement**/**BlockStatement[@CurlyBrace= **false**()]

|

**//**ForEachStatement**/**BlockStatement[@CurlyBrace= **false**()]

**Example(s):**

**for** **(int** i **=** 0**;** i **<** 42**;** i**++)** *// not recommended*

foo**();**

**for** **(int** i **=** 0**;** i **<** 42**;** i**++)** **{** *// preferred approach*

foo**();**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/ForLoopsMustUseBraces" />

FormalParameterNamingConventions

**Since:** PMD 6.15.0

**Priority:** High (1)

Configurable naming conventions for formal parameters of methods. This rule reports formal parameters which do not match the regex that applies to their specific kind (e.g. method parameter, or final method parameter). Each regex can be configured through properties.

By default this rule uses the standard Apex naming convention (Camel case).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.codestyle.FormalParameterNamingConventionsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/codestyle/FormalParameterNamingConventionsRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **bar(Integer** methodParameter**)** **{** **}** *// This is in camel case, so it's ok*

**public** **baz(Integer** METHOD\_PARAMETER**)** **{** **}** *// This will be reported unless you change the regex*

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| finalMethodParameterPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to final method parameter names | no |
| methodParameterPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to method parameter names | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/FormalParameterNamingConventions" />

**Use this rule and customize it:**

<rule ref="category/apex/codestyle.xml/FormalParameterNamingConventions">

<properties>

<property name="finalMethodParameterPattern" value="[a-z][a-zA-Z0-9]\*" />

<property name="methodParameterPattern" value="[a-z][a-zA-Z0-9]\*" />

</properties>

</rule>

IfElseStmtsMustUseBraces

**Since:** PMD 5.6.0

**Priority:** Medium (3)

Avoid using if..else statements without using surrounding braces. If the code formatting or indentation is lost then it becomes difficult to separate the code being controlled from the rest.

**This rule is defined by the following XPath expression:**

**//**IfBlockStatement**/**BlockStatement[@CurlyBrace= **false**()][**count**(child::\*) **>** 0]

|

**//**IfElseBlockStatement**/**BlockStatement[@CurlyBrace= **false**()][**count**(child::\*) **>** 0]

**Example(s):**

*// this is OK*

**if** **(**foo**)** x**++;**

*// but this is not*

**if** **(**foo**)**

x **=** x**+**1**;**

**else**

x **=** x**-**1**;**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/IfElseStmtsMustUseBraces" />

IfStmtsMustUseBraces

**Since:** PMD 5.6.0

**Priority:** Medium (3)

Avoid using if statements without using braces to surround the code block. If the code formatting or indentation is lost then it becomes difficult to separate the code being controlled from the rest.

**This rule is defined by the following XPath expression:**

**//**IfBlockStatement**/**BlockStatement[@CurlyBrace= **false**()]

**Example(s):**

**if** **(**foo**)** *// not recommended*

x**++;**

**if** **(**foo**)** **{** *// preferred approach*

x**++;**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/IfStmtsMustUseBraces" />

LocalVariableNamingConventions

**Since:** PMD 6.15.0

**Priority:** High (1)

Configurable naming conventions for local variable declarations. This rule reports variable declarations which do not match the regex that applies to their specific kind (e.g. local variable, or final local variable). Each regex can be configured through properties.

By default this rule uses the standard Apex naming convention (Camel case).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.codestyle.LocalVariableNamingConventionsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/codestyle/LocalVariableNamingConventionsRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **Foo()** **{**

**Integer** localVariable**;** *// This is in camel case, so it's ok*

**Integer** LOCAL\_VARIABLE**;** *// This will be reported unless you change the regex*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| finalLocalPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to final local variable names | no |
| localPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to local variable names | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/LocalVariableNamingConventions" />

**Use this rule and customize it:**

<rule ref="category/apex/codestyle.xml/LocalVariableNamingConventions">

<properties>

<property name="finalLocalPattern" value="[a-z][a-zA-Z0-9]\*" />

<property name="localPattern" value="[a-z][a-zA-Z0-9]\*" />

</properties>

</rule>

MethodNamingConventions

**Since:** PMD 5.5.0

**Priority:** High (1)

Configurable naming conventions for method declarations. This rule reports method declarations which do not match the regex that applies to their specific kind (e.g. static method, or test method). Each regex can be configured through properties.

By default this rule uses the standard Apex naming convention (Camel case).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.codestyle.MethodNamingConventionsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/codestyle/MethodNamingConventionsRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **void** **instanceMethod()** **{** **}** *// This is in camel case, so it's ok*

**public** **void** **INSTANCE\_METHOD()** **{** **}** *// This will be reported unless you change the regex*

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| skipTestMethodUnderscores | false | Deprecated Skip underscores in test methods | no |
| testPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to test method names | no |
| staticPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to static method names | no |
| instancePattern | [a-z][a-zA-Z0-9]\* | Regex which applies to instance method names | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/MethodNamingConventions" />

**Use this rule and customize it:**

<rule ref="category/apex/codestyle.xml/MethodNamingConventions">

<properties>

<property name="testPattern" value="[a-z][a-zA-Z0-9]\*" />

<property name="staticPattern" value="[a-z][a-zA-Z0-9]\*" />

<property name="instancePattern" value="[a-z][a-zA-Z0-9]\*" />

</properties>

</rule>

OneDeclarationPerLine

**Since:** PMD 6.7.0

**Priority:** High (1)

Apex allows the use of several variables declaration of the same type on one line. However, it can lead to quite messy code. This rule looks for several declarations on the same line.

**This rule is defined by the following XPath expression:**

**//**VariableDeclarationStatements

[**count**(VariableDeclaration) **>** 1 **and** ($reportInForLoopInitializer = **true**() **or** **name**(parent::\*) != 'ForLoopStatement')]

[$strictMode **or** **count**(**distinct-values**(VariableDeclaration/@BeginLine)) != **count**(VariableDeclaration)]

|

**//**FieldDeclarationStatements

[**count**(FieldDeclaration) **>** 1]

[$strictMode **or** **count**(**distinct-values**(FieldDeclaration**/**VariableExpression/@BeginLine)) != **count**(FieldDeclaration**/**VariableExpression)]

**Example(s):**

**Integer** a**,** b**;** *// not recommended*

**Integer** a**,**

b**;** *// ok by default, can be flagged setting the strictMode property*

**Integer** a**;** *// preferred approach*

**Integer** b**;**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| reportInForLoopInitializer | true | If false, multiple declarations in a for loop initializer are not flagged. | no |
| strictMode | false | If true, mark combined declaration even if the declarations are on separate lines. | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/OneDeclarationPerLine" />

**Use this rule and customize it:**

<rule ref="category/apex/codestyle.xml/OneDeclarationPerLine">

<properties>

<property name="reportInForLoopInitializer" value="true" />

<property name="strictMode" value="false" />

</properties>

</rule>

PropertyNamingConventions

**Since:** PMD 6.15.0

**Priority:** High (1)

Configurable naming conventions for property declarations. This rule reports property declarations which do not match the regex that applies to their specific kind (e.g. static property, or instance property). Each regex can be configured through properties.

By default this rule uses the standard Apex naming convention (Camel case).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.codestyle.PropertyNamingConventionsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/codestyle/PropertyNamingConventionsRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **Integer** instanceProperty **{** get**;** set**;** **}** *// This is in camel case, so it's ok*

**public** **Integer** INSTANCE\_PROPERTY **{** get**;** set**;** **}** *// This will be reported unless you change the regex*

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| staticPattern | [a-z][a-zA-Z0-9]\* | Regex which applies to static property names | no |
| instancePattern | [a-z][a-zA-Z0-9]\* | Regex which applies to instance property names | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/PropertyNamingConventions" />

**Use this rule and customize it:**

<rule ref="category/apex/codestyle.xml/PropertyNamingConventions">

<properties>

<property name="staticPattern" value="[a-z][a-zA-Z0-9]\*" />

<property name="instancePattern" value="[a-z][a-zA-Z0-9]\*" />

</properties>

</rule>

VariableNamingConventions

Deprecated

**Since:** PMD 5.5.0

**Priority:** High (1)

A variable naming conventions rule - customize this to your liking. Currently, it checks for final variables that should be fully capitalized and non-final variables that should not include underscores.

This rule is deprecated and will be removed with PMD 7.0.0. The rule is replaced by the more general rules [FieldNamingConventions](https://pmd.github.io/latest/pmd_rules_apex_codestyle.html#fieldnamingconventions), [FormalParameterNamingConventions](https://pmd.github.io/latest/pmd_rules_apex_codestyle.html#formalparameternamingconventions), [LocalVariableNamingConventions](https://pmd.github.io/latest/pmd_rules_apex_codestyle.html#localvariablenamingconventions), and [PropertyNamingConventions](https://pmd.github.io/latest/pmd_rules_apex_codestyle.html" \l "propertynamingconventions).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.codestyle.VariableNamingConventionsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/codestyle/VariableNamingConventionsRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **static** **final** **Integer** MY\_NUM **=** 0**;**

**public** **String** myTest **=** ''**;**

**DataModule** dmTest **=** **new** **DataModule();**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 5 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| checkMembers | true | Check member variables | no |
| checkLocals | true | Check local variables | no |
| checkParameters | true | Check constructor and method parameter variables | no |
| staticPrefix |  | Static variable prefixes | yes. Delimiter is ‘,’. |
| staticSuffix |  | Static variable suffixes | yes. Delimiter is ‘,’. |
| memberPrefix |  | Member variable prefixes | yes. Delimiter is ‘,’. |
| memberSuffix |  | Member variable suffixes | yes. Delimiter is ‘,’. |
| localPrefix |  | Local variable prefixes | yes. Delimiter is ‘,’. |
| localSuffix |  | Local variable suffixes | yes. Delimiter is ‘,’. |
| parameterPrefix |  | Method parameter variable prefixes | yes. Delimiter is ‘,’. |
| parameterSuffix |  | Method parameter variable suffixes | yes. Delimiter is ‘,’. |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/VariableNamingConventions" />

**Use this rule and customize it:**

<rule ref="category/apex/codestyle.xml/VariableNamingConventions">

<properties>

<property name="checkMembers" value="true" />

<property name="checkLocals" value="true" />

<property name="checkParameters" value="true" />

<property name="staticPrefix" value="" />

<property name="staticSuffix" value="" />

<property name="memberPrefix" value="" />

<property name="memberSuffix" value="" />

<property name="localPrefix" value="" />

<property name="localSuffix" value="" />

<property name="parameterPrefix" value="" />

<property name="parameterSuffix" value="" />

</properties>

</rule>

WhileLoopsMustUseBraces

**Since:** PMD 5.6.0

**Priority:** Medium (3)

Avoid using ‘while’ statements without using braces to surround the code block. If the code formatting or indentation is lost then it becomes difficult to separate the code being controlled from the rest.

**This rule is defined by the following XPath expression:**

**//**WhileLoopStatement**/**BlockStatement[@CurlyBrace= **false**()]

**Example(s):**

**while** **(true)** *// not recommended*

x**++;**

**while** **(true)** **{** *// preferred approach*

x**++;**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/codestyle.xml/WhileLoopsMustUseBraces" />

Design

Rules that help you discover design issues.

AvoidDeeplyNestedIfStmts

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Avoid creating deeply nested if-then statements since they are harder to read and error-prone to maintain.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.AvoidDeeplyNestedIfStmtsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/AvoidDeeplyNestedIfStmtsRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **void** **bar(Integer** x**,** **Integer** y**,** **Integer** z**)** **{**

**if** **(**x**>**y**)** **{**

**if** **(**y**>**z**)** **{**

**if** **(**z**==**x**)** **{**

*// !! too deep*

**}**

**}**

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 200 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| problemDepth | 3 | The if statement depth reporting threshold | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/AvoidDeeplyNestedIfStmts" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/AvoidDeeplyNestedIfStmts">

<properties>

<property name="problemDepth" value="3" />

</properties>

</rule>

CognitiveComplexity

**Since:** PMD 6.22.0

**Priority:** Medium (3)

Methods that are highly complex are difficult to read and more costly to maintain. If you include too much decisional logic within a single method, you make its behavior hard to understand and more difficult to modify.

Cognitive complexity is a measure of how difficult it is for humans to read and understand a method. Code that contains a break in the control flow is more complex, whereas the use of language shorthands doesn’t increase the level of complexity. Nested control flows can make a method more difficult to understand, with each additional nesting of the control flow leading to an increase in cognitive complexity.

Information about Cognitive complexity can be found in the original paper here: <https://www.sonarsource.com/docs/CognitiveComplexity.pdf>

By default, this rule reports methods with a complexity of 15 or more. Reported methods should be broken down into less complex components.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.CognitiveComplexityRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/CognitiveComplexityRule.java)

**Example(s):**

**public** **class** **Foo** **{**

*// Has a cognitive complexity of 0*

**public** **void** **createAccount()** **{**

**Account** account **=** **new** **Account(Name** **=** 'PMD'**);**

insert account**;**

**}**

*// Has a cognitive complexity of 1*

**public** **Boolean** **setPhoneNumberIfNotExisting(Account** a**,** **String** phone**)** **{**

**if** **(**a**.**Phone **==** **null)** **{** *// +1*

a**.**Phone **=** phone**;**

update a**;**

**return** **true;**

**}**

**return** **false;**

**}**

*// Has a cognitive complexity of 4*

**public** **void** **updateContacts(List<Contact>** contacts**)** **{**

**List<Contact>** contactsToUpdate **=** **new** **List<Contact>();**

**for** **(Contact** contact **:** contacts**)** **{** *// +1*

**if** **(**contact**.**Department **==** '**Finance**'**)** **{** *// +2 (nesting = 1)*

contact**.**Title **=** '**Finance** **Specialist**'**;**

contactsToUpdate**.**add**(**contact**);**

**}** **else** **if** **(**contact**.**Department **==** '**Sales**'**)** **{** *// +1*

contact**.**Title **=** '**Sales** **Specialist**'**;**

contactsToUpdate**.**add**(**contact**);**

**}**

**}**

update contactsToUpdate**;**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| classReportLevel | 50 | Total class cognitive complexity reporting threshold | no |
| methodReportLevel | 15 | Cognitive complexity reporting threshold | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/CognitiveComplexity" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/CognitiveComplexity">

<properties>

<property name="classReportLevel" value="50" />

<property name="methodReportLevel" value="15" />

</properties>

</rule>

CyclomaticComplexity

**Since:** PMD 6.0.0

**Priority:** Medium (3)

The complexity of methods directly affects maintenance costs and readability. Concentrating too much decisional logic in a single method makes its behaviour hard to read and change.

Cyclomatic complexity assesses the complexity of a method by counting the number of decision points in a method, plus one for the method entry. Decision points are places where the control flow jumps to another place in the program. As such, they include all control flow statements, such as ‘if’, ‘while’, ‘for’, and ‘case’.

Generally, numbers ranging from 1-4 denote low complexity, 5-7 denote moderate complexity, 8-10 denote high complexity, and 11+ is very high complexity. By default, this rule reports methods with a complexity >= 10. Additionally, classes with many methods of moderate complexity get reported as well once the total of their methods’ complexities reaches 40, even if none of the methods was directly reported.

Reported methods should be broken down into several smaller methods. Reported classes should probably be broken down into subcomponents.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.CyclomaticComplexityRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/CyclomaticComplexityRule.java)

**Example(s):**

**public** **class** **Complicated** **{**

**public** **void** **example()** **{** *// This method has a cyclomatic complexity of 12*

**int** x **=** 0**,** y **=** 1**,** z **=** 2**,** t **=** 2**;**

**boolean** a **=** **false,** b **=** **true,** c **=** **false,** d **=** **true;**

**if** **(**a **&&** b **||** b **&&** d**)** **{**

**if** **(**y **==** z**)** **{**

x **=** 2**;**

**}** **else** **if** **(**y **==** t **&&** **!**d**)** **{**

x **=** 2**;**

**}** **else** **{**

x **=** 2**;**

**}**

**}** **else** **if** **(**c **&&** d**)** **{**

**while** **(**z **<** y**)** **{**

x **=** 2**;**

**}**

**}** **else** **{**

**for** **(int** n **=** 0**;** n **<** t**;** n**++)** **{**

x **=** 2**;**

**}**

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| classReportLevel | 40 | Total class complexity reporting threshold | no |
| methodReportLevel | 10 | Cyclomatic complexity reporting threshold | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/CyclomaticComplexity" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/CyclomaticComplexity">

<properties>

<property name="classReportLevel" value="40" />

<property name="methodReportLevel" value="10" />

</properties>

</rule>

ExcessiveClassLength

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Excessive class file lengths are usually indications that the class may be burdened with excessive responsibilities that could be provided by external classes or functions. In breaking these methods apart the code becomes more managable and ripe for reuse.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.ExcessiveClassLengthRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/ExcessiveClassLengthRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **void** **bar1()** **{**

*// 1000 lines of code*

**}**

**public** **void** **bar2()** **{**

*// 1000 lines of code*

**}**

**public** **void** **bar3()** **{**

*// 1000 lines of code*

**}**

**public** **void** **barN()** **{**

*// 1000 lines of code*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| topscore |  | Deprecated Top score value | no |
| minimum | 1000.0 | Minimum reporting threshold | no |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 150 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| sigma |  | Deprecated Sigma value | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/ExcessiveClassLength" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/ExcessiveClassLength">

<properties>

<property name="minimum" value="1000.0" />

</properties>

</rule>

ExcessiveParameterList

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Methods with numerous parameters are a challenge to maintain, especially if most of them share the same datatype. These situations usually denote the need for new objects to wrap the numerous parameters.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.ExcessiveParameterListRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/ExcessiveParameterListRule.java)

**Example(s):**

*// too many arguments liable to be mixed up*

**public** **void** **addPerson(Integer** birthYear**,** **Integer** birthMonth**,** **Integer** birthDate**,** **Integer** height**,** **Integer** weight**,** **Integer** ssn**)** **{**

*// ...*

**}**

*// preferred approach*

**public** **void** **addPerson(Date** birthdate**,** **BodyMeasurements** measurements**,** **int** ssn**)** **{**

*// ...*

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| topscore |  | Deprecated Top score value | no |
| minimum | 4.0 | Minimum reporting threshold | no |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 50 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| sigma |  | Deprecated Sigma value | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/ExcessiveParameterList" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/ExcessiveParameterList">

<properties>

<property name="minimum" value="4.0" />

</properties>

</rule>

ExcessivePublicCount

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Classes with large numbers of public methods and attributes require disproportionate testing efforts since combinational side effects grow rapidly and increase risk. Refactoring these classes into smaller ones not only increases testability and reliability but also allows new variations to be developed easily.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.ExcessivePublicCountRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/ExcessivePublicCountRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **String** value**;**

**public** **Bar** something**;**

**public** **Variable** **var;**

*// [... more more public attributes ...]*

**public** **void** **doWork()** **{}**

**public** **void** **doMoreWork()** **{}**

**public** **void** **doWorkAgain()** **{}**

*// [... more more public methods ...]*

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| topscore |  | Deprecated Top score value | no |
| minimum | 20.0 | Minimum reporting threshold | no |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 150 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| sigma |  | Deprecated Sigma value | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/ExcessivePublicCount" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/ExcessivePublicCount">

<properties>

<property name="minimum" value="20.0" />

</properties>

</rule>

NcssConstructorCount

**Since:** PMD 5.5.0

**Priority:** Medium (3)

This rule uses the NCSS (Non-Commenting Source Statements) algorithm to determine the number of lines of code for a given constructor. NCSS ignores comments, and counts actual statements. Using this algorithm, lines of code that are split are counted as one.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.NcssConstructorCountRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/NcssConstructorCountRule.java)

**Example(s):**

**public** **class** **Foo** **extends** **Bar** **{**

*//this constructor only has 1 NCSS lines*

**public** **Foo()** **{**

**super();**

**super.**foo**();**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| topscore |  | Deprecated Top score value | no |
| minimum | 20.0 | Minimum reporting threshold | no |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 50 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| sigma |  | Deprecated Sigma value | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/NcssConstructorCount" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/NcssConstructorCount">

<properties>

<property name="minimum" value="20.0" />

</properties>

</rule>

NcssMethodCount

**Since:** PMD 5.5.0

**Priority:** Medium (3)

This rule uses the NCSS (Non-Commenting Source Statements) algorithm to determine the number of lines of code for a given method. NCSS ignores comments, and counts actual statements. Using this algorithm, lines of code that are split are counted as one.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.NcssMethodCountRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/NcssMethodCountRule.java)

**Example(s):**

**public** **class** **Foo** **extends** **Bar** **{**

*//this method only has 1 NCSS lines*

**public** **Integer** **method()** **{**

**super.**method**();**

**return** 1**;**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| topscore |  | Deprecated Top score value | no |
| minimum | 40.0 | Minimum reporting threshold | no |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 50 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| sigma |  | Deprecated Sigma value | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/NcssMethodCount" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/NcssMethodCount">

<properties>

<property name="minimum" value="40.0" />

</properties>

</rule>

NcssTypeCount

**Since:** PMD 5.5.0

**Priority:** Medium (3)

This rule uses the NCSS (Non-Commenting Source Statements) algorithm to determine the number of lines of code for a given type. NCSS ignores comments, and counts actual statements. Using this algorithm, lines of code that are split are counted as one.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.NcssTypeCountRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/NcssTypeCountRule.java)

**Example(s):**

*//this class only has 6 NCSS lines*

**public** **class** **Foo** **extends** **Bar** **{**

**public** **Foo()** **{**

**super();**

**super.**foo**();**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| topscore |  | Deprecated Top score value | no |
| minimum | 500.0 | Minimum reporting threshold | no |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 250 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| sigma |  | Deprecated Sigma value | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/NcssTypeCount" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/NcssTypeCount">

<properties>

<property name="minimum" value="500.0" />

</properties>

</rule>

StdCyclomaticComplexity

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Complexity directly affects maintenance costs is determined by the number of decision points in a method plus one for the method entry. The decision points include ‘if’, ‘while’, ‘for’, and ‘case labels’ calls. Generally, numbers ranging from 1-4 denote low complexity, 5-7 denote moderate complexity, 8-10 denote high complexity, and 11+ is very high complexity.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.StdCyclomaticComplexityRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/StdCyclomaticComplexityRule.java)

**Example(s):**

*// This has a Cyclomatic Complexity = 12*

**public** **class** **Foo** **{**

1 **public** **void** **example()** **{**

2 **if** **(**a **==** b **||** **(**c **==** d **&&** e **==** f**))** **{**

3 **if** **(**a1 **==** b1**)** **{**

fiddle**();**

4 **}** **else** **if** a2 **==** b2**)** **{**

fiddle**();**

**}** **else** **{**

fiddle**();**

**}**

5 **}** **else** **if** **(**c **==** d**)** **{**

6 **while** **(**c **==** d**)** **{**

fiddle**();**

**}**

7 **}** **else** **if** **(**e **==** f**)** **{**

8 **for** **(int** n **=** 0**;** n **<** h**;** n**++)** **{**

fiddle**();**

**}**

**}** **else** **{**

**switch** **(**z**)** **{**

9 **case** 1**:**

fiddle**();**

**break;**

10 **case** 2**:**

fiddle**();**

**break;**

11 **case** 3**:**

fiddle**();**

**break;**

12 **default:**

fiddle**();**

**break;**

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 250 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| reportLevel | 10 | Cyclomatic Complexity reporting threshold | no |
| showClassesComplexity | true | Add class average violations to the report | no |
| showMethodsComplexity | true | Add method average violations to the report | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/StdCyclomaticComplexity" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/StdCyclomaticComplexity">

<properties>

<property name="reportLevel" value="10" />

<property name="showClassesComplexity" value="true" />

<property name="showMethodsComplexity" value="true" />

</properties>

</rule>

TooManyFields

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Classes that have too many fields can become unwieldy and could be redesigned to have fewer fields, possibly through grouping related fields in new objects. For example, a class with individual city/state/zip fields could park them within a single Address field.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.design.TooManyFieldsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/design/TooManyFieldsRule.java)

**Example(s):**

**public** **class** **Person** **{**

*// too many separate fields*

**Integer** birthYear**;**

**Integer** birthMonth**;**

**Integer** birthDate**;**

**Double** height**;**

**Double** weight**;**

**}**

**public** **class** **Person** **{**

*// this is more manageable*

**Date** birthDate**;**

**BodyMeasurements** measurements**;**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Complexity | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 200 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| maxfields | 15 | Max allowable fields | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/design.xml/TooManyFields" />

**Use this rule and customize it:**

<rule ref="category/apex/design.xml/TooManyFields">

<properties>

<property name="maxfields" value="15" />

</properties>

</rule>

Documentation

Rules that are related to code documentation.

ApexDoc

**Since:** PMD 6.8.0

**Priority:** Medium (3)

This rule validates that:

* ApexDoc comments are present for classes, methods, and properties that are public or global, excluding overrides and test classes (as well as the contents of test classes).
* ApexDoc comments are present for classes, methods, and properties that are protected or private, depending on the properties reportPrivate and reportProtected.
* ApexDoc comments should contain @description depending on the property reportMissingDescription.
* ApexDoc comments on non-void, non-constructor methods should contain @return.
* ApexDoc comments on void or constructor methods should not contain @return.
* ApexDoc comments on methods with parameters should contain @param for each parameter, in the same order as the method signature.
* ApexDoc comments are present on properties is only validated, if the property reportProperty is enabled. By setting reportProperty to false, you can ignore missing comments on properties.

Method overrides and tests are both exempted from having ApexDoc.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.documentation.ApexDocRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/documentation/ApexDocRule.java)

**Example(s):**

*/\*\**

*\* @description Hello World*

*\*/*

**public** **class** **HelloWorld** **{**

*/\*\**

*\* @description Bar*

*\* @return Bar*

*\*/*

**public** **Object** **bar()** **{** **return** **null;** **}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| reportPrivate | false | Report private classes, methods and properties | no |
| reportProtected | false | Report protected classes, methods and properties | no |
| reportMissingDescription | true | Report missing @description | no |
| reportProperty | true | Report properties without comments | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/documentation.xml/ApexDoc" />

**Use this rule and customize it:**

<rule ref="category/apex/documentation.xml/ApexDoc">

<properties>

<property name="reportPrivate" value="false" />

<property name="reportProtected" value="false" />

<property name="reportMissingDescription" value="true" />

<property name="reportProperty" value="true" />

</properties>

</rule>

Error Prone

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

ApexCSRF

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Having DML operations in Apex class constructor or initializers can have unexpected side effects: By just accessing a page, the DML statements would be executed and the database would be modified. Just querying the database is permitted.

In addition to constructors and initializers, any method called init is checked as well.

Salesforce Apex already protects against this scenario and raises a runtime exception.

Note: This rule has been moved from category "Security" to "Error Prone" with PMD 6.21.0, since using DML in constructors is not a security problem, but crashes the application.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.errorprone.ApexCSRFRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/errorprone/ApexCSRFRule.java)

**Example(s):**

**public** **class** **Foo** **{**

*// initializer*

**{**

insert data**;**

**}**

*// static initializer*

**static** **{**

insert data**;**

**}**

*// constructor*

**public** **Foo()** **{**

insert data**;**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/ApexCSRF" />

AvoidDirectAccessTriggerMap

**Since:** PMD 6.0.0

**Priority:** Medium (3)

Avoid directly accessing Trigger.old and Trigger.new as it can lead to a bug. Triggers should be bulkified and iterate through the map to handle the actions for each item separately.

**This rule is defined by the following XPath expression:**

**//**ArrayLoadExpression[TriggerVariableExpression **and** LiteralExpression]

**Example(s):**

trigger **AccountTrigger** on **Account** **(**before insert**,** before update**)** **{**

**Account** a **=** **Trigger.**new**[**0**];** *//Bad: Accessing the trigger array directly is not recommended.*

**for** **(** **Account** a **:** **Trigger.**new **)** **{**

*//Good: Iterate through the trigger.new array instead.*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/AvoidDirectAccessTriggerMap" />

AvoidHardcodingId

**Since:** PMD 6.0.0

**Priority:** Medium (3)

When deploying Apex code between sandbox and production environments, or installing Force.com AppExchange packages, it is essential to avoid hardcoding IDs in the Apex code. By doing so, if the record IDs change between environments, the logic can dynamically identify the proper data to operate against and not fail.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.errorprone.AvoidHardcodingIdRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/errorprone/AvoidHardcodingIdRule.java)

**Example(s):**

**public** without sharing **class** **Foo** **{**

**void** **foo()** **{**

*//Error - hardcoded the record type id*

**if** **(**a**.**RecordTypeId **==** '012500000009**WAr**'**)** **{**

*//do some logic here.....*

**}** **else** **if** **(**a**.**RecordTypeId **==** '0123000000095**Km**'**)** **{**

*//do some logic here for a different record type...*

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/AvoidHardcodingId" />

AvoidNonExistentAnnotations

**Since:** PMD 6.5.0

**Priority:** Medium (3)

Apex supported non existent annotations for legacy reasons. In the future, use of such non-existent annotations could result in broken apex code that will not compile. This will prevent users of garbage annotations from being able to use legitimate annotations added to Apex in the future. A full list of supported annotations can be found at https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex\_classes\_annotation.htm

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.errorprone.AvoidNonExistentAnnotationsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/errorprone/AvoidNonExistentAnnotationsRule.java)

**Example(s):**

@NonExistentAnnotation **public** **class** **ClassWithNonexistentAnnotation** **{**

@NonExistentAnnotation **public** **void** **methodWithNonExistentAnnotation()** **{**

*// ...*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/AvoidNonExistentAnnotations" />

EmptyCatchBlock

**Since:** PMD 6.0.0

**Priority:** Medium (3)

Empty Catch Block finds instances where an exception is caught, but nothing is done. In most circumstances, this swallows an exception which should either be acted on or reported.

**This rule is defined by the following XPath expression:**

**//**CatchBlockStatement[.**/**BlockStatement[**count**(\*) = 0] **and**

**not**(**matches**(@VariableName, $allowExceptionNameRegex)) **and**

($allowCommentedBlocks = **false**() **or** @ContainsComment = **false**())]

**Example(s):**

**public** **void** **doSomething()** **{**

**...**

**try** **{**

insert accounts**;**

**}** **catch** **(DmlException** dmle**)** **{**

*// not good*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| allowCommentedBlocks | false | Empty blocks containing comments will be skipped | no |
| allowExceptionNameRegex | ^(ignored|expected)$ | Empty blocks catching exceptions with names matching this regular expression will be skipped | no |

**Use this rule with the default properties by just referencing it:**

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

**Use this rule and customize it:**

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

<properties>

<property name="allowCommentedBlocks" value="false" />

<property name="allowExceptionNameRegex" value="^(ignored|expected)$" />

</properties>

</rule>

EmptyIfStmt

**Since:** PMD 6.0.0

**Priority:** Medium (3)

Empty If Statement finds instances where a condition is checked but nothing is done about it.

**This rule is defined by the following XPath expression:**

**//**IfBlockStatement

[BlockStatement[**count**(\*) = 0]]

**Example(s):**

**public** **class** **Foo** **{**

**public** **void** **bar(Integer** x**)** **{**

**if** **(**x **==** 0**)** **{**

*// empty!*

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/EmptyIfStmt" />

EmptyStatementBlock

**Since:** PMD 6.0.0

**Priority:** Medium (3)

Empty block statements serve no purpose and should be removed.

**This rule is defined by the following XPath expression:**

**//**Method[$reportEmptyPrivateNoArgConstructor = **true**() **or** (@Constructor != **true**() **or** .**/**ModifierNode[@Private != **true**()] **or** .**/**Parameter[**count**(\*) **>** 0])]**/**ModifierNode[@Abstract != **true**() **and** ($reportEmptyVirtualMethod = **true**() **or** @Virtual != **true**()) **and** ..**/**BlockStatement[**count**(\*) = 0]]

| **//**Method**/**BlockStatement**//**BlockStatement[**count**(\*) = 0 **and** @Location != parent::\*/@Location]

**Example(s):**

**public** **class** **Foo** **{**

**private** **Integer** \_bar**;**

**public** **void** **setBar(Integer** bar**)** **{**

*// empty*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| reportEmptyPrivateNoArgConstructor | true | If false, empty private no-arg constructors are not flagged. This supports a common idiom used by singleton pattern implementations, utility classes, etc. | no |
| reportEmptyVirtualMethod | true | If false, empty virtual methods are not flagged. This supports abstract base classes with default no-op implementations. | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/EmptyStatementBlock" />

**Use this rule and customize it:**

<rule ref="category/apex/errorprone.xml/EmptyStatementBlock">

<properties>

<property name="reportEmptyPrivateNoArgConstructor" value="true" />

<property name="reportEmptyVirtualMethod" value="true" />

</properties>

</rule>

EmptyTryOrFinallyBlock

**Since:** PMD 6.0.0

**Priority:** Medium (3)

Avoid empty try or finally blocks - what’s the point?

**This rule is defined by the following XPath expression:**

**//**TryCatchFinallyBlockStatement[.**/**BlockStatement[**count**(\*) = 0]]

**Example(s):**

**public** **class** **Foo** **{**

**public** **void** **bar()** **{**

**try** **{**

*// empty !*

**}** **catch** **(Exception** e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**public** **class** **Foo** **{**

**public** **void** **bar()** **{**

**try** **{**

**Integer** x**=**2**;**

**}** **finally** **{**

*// empty!*

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/EmptyTryOrFinallyBlock" />

EmptyWhileStmt

**Since:** PMD 6.0.0

**Priority:** Medium (3)

Empty While Statement finds all instances where a while statement does nothing. If it is a timing loop, then you should use Thread.sleep() for it; if it is a while loop that does a lot in the exit expression, rewrite it to make it clearer.

**This rule is defined by the following XPath expression:**

**//**WhileLoopStatement[.**/**BlockStatement[**count**(\*) = 0]]

**Example(s):**

**public** **void** **bar(Integer** a**,** **Integer** b**)** **{**

**while** **(**a **==** b**)** **{**

*// empty!*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/EmptyWhileStmt" />

InaccessibleAuraEnabledGetter

**Since:** PMD 6.36.0

**Priority:** Medium (3)

In the Summer ‘21 release, a mandatory security update enforces access modifiers on Apex properties in Lightning component markup. The update prevents access to private or protected Apex getters from Aura and Lightning Web Components.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.errorprone.InaccessibleAuraEnabledGetterRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/errorprone/InaccessibleAuraEnabledGetterRule.java)

**Example(s):**

**public** **class** **Foo** **{**

@AuraEnabled

**public** **Integer** counter **{** **private** get**;** set**;** **}** *// Violating - Private getter is inaccessible to Lightning components*

@AuraEnabled

**public** **static** **Foo** **bar()**

**{**

**Foo** foo **=** **new** **Foo();**

foo**.**counter **=** 2**;**

**return** foo**;**

**}**

**}**

**public** **class** **Foo** **{**

@AuraEnabled

**public** **Integer** counter **{** **protected** get**;** set**;** **}** *// Violating - Protected getter is inaccessible to Lightning components*

@AuraEnabled

**public** **static** **Foo** **bar()**

**{**

**Foo** foo **=** **new** **Foo();**

foo**.**counter **=** 2**;**

**return** foo**;**

**}**

**}**

**public** **class** **Foo** **{**

@AuraEnabled

**public** **Integer** counter **{** get**;** set**;** **}** *// Compliant - Public getter is accessible to Lightning components*

@AuraEnabled

**public** **static** **Foo** **bar()**

**{**

**Foo** foo **=** **new** **Foo();**

foo**.**counter **=** 2**;**

**return** foo**;**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/InaccessibleAuraEnabledGetter" />

MethodWithSameNameAsEnclosingClass

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Non-constructor methods should not have the same name as the enclosing class.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.errorprone.MethodWithSameNameAsEnclosingClassRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/errorprone/MethodWithSameNameAsEnclosingClassRule.java)

**Example(s):**

**public** **class** **MyClass** **{**

*// this is OK because it is a constructor*

**public** **MyClass()** **{}**

*// this is bad because it is a method*

**public** **void** **MyClass()** **{}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 50 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/MethodWithSameNameAsEnclosingClass" />

OverrideBothEqualsAndHashcode

**Since:** PMD 6.31.0

**Priority:** Medium (3)

Override both public Boolean equals(Object obj), and public Integer hashCode(), or override neither. Even if you are inheriting a hashCode() from a parent class, consider implementing hashCode and explicitly delegating to your superclass.

This is especially important when [Using Custom Types in Map Keys and Sets](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_collections_maps_keys_userdefined.htm).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.errorprone.OverrideBothEqualsAndHashcodeRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/errorprone/OverrideBothEqualsAndHashcodeRule.java)

**Example(s):**

**public** **class** **Bar** **{** *// poor, missing a hashCode() method*

**public** **Boolean** **equals(Object** o**)** **{**

*// do some comparison*

**}**

**}**

**public** **class** **Baz** **{** *// poor, missing an equals() method*

**public** **Integer** **hashCode()** **{**

*// return some hash value*

**}**

**}**

**public** **class** **Foo** **{** *// perfect, both methods provided*

**public** **Boolean** **equals(Object** other**)** **{**

*// do some comparison*

**}**

**public** **Integer** **hashCode()** **{**

*// return some hash value*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/OverrideBothEqualsAndHashcode" />

TestMethodsMustBeInTestClasses

**Since:** PMD 6.22.0

**Priority:** Medium (3)

Test methods marked as a testMethod or annotated with @IsTest, but not residing in a test class should be moved to a proper class or have the @IsTest annotation added to the class.

Support for tests inside functional classes was removed in Spring-13 (API Version 27.0), making classes that violate this rule fail compile-time. This rule is mostly usable when dealing with legacy code.

**This rule is defined by the following XPath expression:**

**//**UserClass[

**not**(.**/**ModifierNode**/**Annotation[**lower-case**(@Image) = 'istest']) **and**

(

(.**/**Method**/**ModifierNode**/**Annotation[**lower-case**(@Image) = 'istest']) **or**

(.**/**Method**/**ModifierNode[@Test = **true**()])

)

]

**Example(s):**

*// Violating*

**private** **class** **TestClass** **{**

@IsTest **static** **void** **myTest()** **{**

*// Code here*

**}**

**}**

**private** **class** **TestClass** **{**

**static** testMethod **void** **myTest()** **{**

*// Code here*

**}**

**}**

*// Compliant*

@IsTest

**private** **class** **TestClass** **{**

@IsTest **static** **void** **myTest()** **{**

*// Code here*

**}**

**}**

@IsTest

**private** **class** **TestClass** **{**

**static** testMethod **void** **myTest()** **{**

*// Code here*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/errorprone.xml/TestMethodsMustBeInTestClasses" />

Performance

Rules that flag suboptimal code.

AvoidDebugStatements

**Since:** PMD 6.36.0

**Priority:** Medium (3)

Debug statements contribute to longer transactions and consume Apex CPU time even when debug logs are not being captured.

When possible make use of other debugging techniques such as the Apex Replay Debugger and Checkpoints that could cover *most* use cases.

For other valid use cases that the statement is in fact valid make use of the @SuppressWarnings annotation or the //NOPMD comment.

**This rule is defined by the following XPath expression:**

**//**MethodCallExpression[**lower-case**(@FullMethodName)='system.debug']

**Example(s):**

**public** **class** **Foo** **{**

**public** **void** **bar()** **{**

**Account** acc **=** **[**SELECT **Name,** **Owner.**Name FROM **Account** LIMIT 1**];**

**System.**debug**(**accs**);** *// will get reported*

**}**

@SuppressWarnings**(**'PMD**.**AvoidDebugStatements'**)**

**public** **void** **baz()** **{**

**try** **{**

**Account** myAccount **=** bar**();**

**}** **catch** **(Exception** e**)** **{**

**System.**debug**(LoggingLevel.**ERROR**,** e**.**getMessage**());** *// good to go*

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/performance.xml/AvoidDebugStatements" />

AvoidDmlStatementsInLoops

Deprecated

**Since:** PMD 5.5.0

**Priority:** Medium (3)

Avoid DML statements inside loops to avoid hitting the DML governor limit. Instead, try to batch up the data into a list and invoke your DML once on that list of data outside the loop.

This rule is deprecated and will be removed with PMD 7.0.0. The rule is replaced by the more general rule [OperationWithLimitsInLoop](https://pmd.github.io/latest/pmd_rules_apex_performance.html" \l "operationwithlimitsinloop).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.performance.AvoidDmlStatementsInLoopsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/performance/AvoidDmlStatementsInLoopsRule.java)

**Example(s):**

**public** **class** **Something** **{**

**public** **void** **foo()** **{**

**for** **(Integer** i **=** 0**;** i **<** 151**;** i**++)** **{**

**Account** account**;**

*// ...*

insert account**;**

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Performance | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 150 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/performance.xml/AvoidDmlStatementsInLoops" />

AvoidSoqlInLoops

Deprecated

**Since:** PMD 5.5.0

**Priority:** Medium (3)

New objects created within loops should be checked to see if they can created outside them and reused.

This rule is deprecated and will be removed with PMD 7.0.0. The rule is replaced by the more general rule [OperationWithLimitsInLoop](https://pmd.github.io/latest/pmd_rules_apex_performance.html" \l "operationwithlimitsinloop).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.performance.AvoidSoqlInLoopsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/performance/AvoidSoqlInLoopsRule.java)

**Example(s):**

**public** **class** **Something** **{**

**public** **static** **void** **main(** **String** as**[]** **)** **{**

**for** **(Integer** i **=** 0**;** i **<** 10**;** i**++)** **{**

**List<Account>** accounts **=** **[**SELECT **Id** FROM **Account];**

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Performance | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 150 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/performance.xml/AvoidSoqlInLoops" />

AvoidSoslInLoops

Deprecated

**Since:** PMD 6.0.0

**Priority:** Medium (3)

Sosl calls within loops can cause governor limit exceptions.

This rule is deprecated and will be removed with PMD 7.0.0. The rule is replaced by the more general rule [OperationWithLimitsInLoop](https://pmd.github.io/latest/pmd_rules_apex_performance.html" \l "operationwithlimitsinloop).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.performance.AvoidSoslInLoopsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/performance/AvoidSoslInLoopsRule.java)

**Example(s):**

**public** **class** **Something** **{**

**public** **static** **void** **main(** **String** as**[]** **)** **{**

**for** **(Integer** i **=** 0**;** i **<** 10**;** i**++)** **{**

**List<List<SObject>>** searchList **=** **[**FIND 'map**\***' IN ALL FIELDS RETURNING **Account** **(Id,** **Name),** **Contact,** **Opportunity,** **Lead];**

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Performance | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 150 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/performance.xml/AvoidSoslInLoops" />

EagerlyLoadedDescribeSObjectResult

**Since:** PMD 6.40.0

**Priority:** Medium (3)

This rule finds DescribeSObjectResults which could have been loaded eagerly via SObjectType.getDescribe().

When using SObjectType.getDescribe() or Schema.describeSObjects() without supplying a SObjectDescribeOptions, implicitly it will be using SObjectDescribeOptions.DEFAULT and then all child relationships will be loaded eagerly regardless whether this information is needed or not. This has a potential negative performance impact. Instead [SObjectType.getDescribe(options)](https://developer.salesforce.com/docs/atlas.en-us.apexref.meta/apexref/apex_class_Schema_SObjectType.htm#unique_346834793) or [Schema.describeSObjects(SObjectTypes, options)](https://developer.salesforce.com/docs/atlas.en-us.apexref.meta/apexref/apex_methods_system_schema.htm#apex_System_Schema_describeSObjects) should be used and a SObjectDescribeOptions should be supplied. By using SObjectDescribeOptions.DEFERRED the describe attributes will be lazily initialized at first use.

Lazy loading DescribeSObjectResult on picklist fields is not always recommended. The lazy loaded describe objects might not be 100% accurate. It might be safer to explicitly use SObjectDescribeOptions.FULL in such a case. The same applies when you need the same DescribeSObjectResult to be consistent across different contexts and API versions.

Properties:

* noDefault: The behavior of SObjectDescribeOptions.DEFAULT changes from API Version 43 to 44: With API Version 43, the attributes are loaded eagerly. With API Version 44, they are loaded lazily. Simply using SObjectDescribeOptions.DEFAULT doesn’t automatically make use of lazy loading. (unless “Use Improved Schema Caching” critical update is applied, SObjectDescribeOptions.DEFAULT does fallback to lazy loading) With this property enabled, such usages are found. You might ignore this, if you can make sure, that you don’t run a mix of API Versions.

**This rule is defined by the following XPath expression:**

**//**MethodCallExpression

[

**lower-case**(@MethodName) = "getdescribe" **and** ReferenceExpression[@SObjectType = **true**()]

**or** **lower-case**(@MethodName) = "describesobjects"

]

[**not**(VariableExpression**/**ReferenceExpression

[**lower-case**(@Image) = ("sobjectdescribeoptions", "fielddescribeoptions")]

)

]

|

**//**ReferenceExpression

[$noDefault = **true**()]

[**lower-case**(@Image) = "sobjectdescribeoptions"]

[parent::VariableExpression[**lower-case**(@Image) = "default"]]

**Example(s):**

**public** **class** **Foo** **{**

**public** **static** **void** **bar(List<Account>** accounts**)** **{**

**if** **(Account.**SObjectType**.**getDescribe**(SObjectDescribeOptions.**DEFERRED**).**isCreateable**())** **{**

insert accounts**;**

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Style | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 1 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| noDefault | false | Do not allow SObjectDescribeOptions.DEFAULT option to ensure consistent results no matter where getDescribe is called | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/performance.xml/EagerlyLoadedDescribeSObjectResult" />

**Use this rule and customize it:**

<rule ref="category/apex/performance.xml/EagerlyLoadedDescribeSObjectResult">

<properties>

<property name="noDefault" value="false" />

</properties>

</rule>

OperationWithLimitsInLoop

**Since:** PMD 6.29.0

**Priority:** Medium (3)

Database class methods, DML operations, SOQL queries, SOSL queries, Approval class methods, Email sending, async scheduling or queueing within loops can cause governor limit exceptions. Instead, try to batch up the data into a list and invoke the operation once on that list of data outside the loop.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.performance.OperationWithLimitsInLoopRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/performance/OperationWithLimitsInLoopRule.java)

**Example(s):**

**public** **class** **Something** **{**

**public** **void** **databaseMethodInsideOfLoop(List<Account>** accounts**)** **{**

**for** **(Account** a **:** accounts**)** **{**

**Database.**insert**(**a**);**

**}**

**}**

**public** **void** **dmlInsideOfLoop()** **{**

**for** **(Integer** i **=** 0**;** i **<** 151**;** i**++)** **{**

**Account** account**;**

*// ...*

insert account**;**

**}**

**}**

**public** **void** **soqlInsideOfLoop()** **{**

**for** **(Integer** i **=** 0**;** i **<** 10**;** i**++)** **{**

**List<Account>** accounts **=** **[**SELECT **Id** FROM **Account];**

**}**

**}**

**public** **void** **soslInsideOfLoop()** **{**

**for** **(Integer** i **=** 0**;** i **<** 10**;** i**++)** **{**

**List<List<SObject>>** searchList **=** **[**FIND 'map**\***' IN ALL FIELDS RETURNING **Account** **(Id,** **Name),** **Contact,** **Opportunity,** **Lead];**

**}**

**}**

**public** **void** **messageInsideOfLoop()** **{**

**for** **(Integer** i **=** 0**;** i **<** 10**;** i**++)** **{**

**Messaging.**SingleEmailMessage email **=** **new** **Messaging.**SingleEmailMessage**();**

**Messaging.**sendEmail**(new** **Messaging.**SingleEmailMessage**[]{**email**});**

**}**

**}**

**public** **void** **approvalInsideOfLoop(Account[]** accs**)** **{**

**for** **(Integer** i **=** 0**;** i **<** 10**;** i**++)** **{**

**Account** acc **=** accs**[**i**];**

**Approval.**ProcessSubmitRequest req **=** **new** **Approval.**ProcessSubmitRequest**();**

req**.**setObjectId**(**acc**.**Id**);**

**Approval.**process**(**req**);**

**Approval.**lock**(**acc**);**

**Approval.**unlock**(**acc**);**

**}**

**}**

**public** **void** **asyncInsideOfLoop()** **{**

**for** **(Integer** i **=** 0**;** i **<** 10**;** i**++)** **{**

**System.**enqueueJob**(new** **MyQueueable());**

**System.**schedule**(**'x'**,** '0 0 0 1 1 **?**'**,** **new** **MySchedule());**

**System.**scheduleBatch**(new** **MyBatch(),** 'x'**,** 1**);**

**}**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Performance | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 150 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/performance.xml/OperationWithLimitsInLoop" />

Security

Rules that flag potential security flaws.

ApexBadCrypto

**Since:** PMD 5.5.3

**Priority:** Medium (3)

The rule makes sure you are using randomly generated IVs and keys for Crypto calls. Hard-wiring these values greatly compromises the security of encrypted data.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexBadCryptoRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexBadCryptoRule.java)

**Example(s):**

**public** without sharing **class** **Foo** **{**

**Blob** hardCodedIV **=** **Blob.**valueOf**(**'**Hardcoded** IV 123'**);**

**Blob** hardCodedKey **=** **Blob.**valueOf**(**'0000000000000000'**);**

**Blob** data **=** **Blob.**valueOf**(**'**Data** to be encrypted'**);**

**Blob** encrypted **=** **Crypto.**encrypt**(**'AES128'**,** hardCodedKey**,** hardCodedIV**,** data**);**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexBadCrypto" />

ApexCRUDViolation

**Since:** PMD 5.5.3

**Priority:** Medium (3)

The rule validates you are checking for access permissions before a SOQL/SOSL/DML operation. Since Apex runs in system mode not having proper permissions checks results in escalation of privilege and may produce runtime errors. This check forces you to handle such scenarios.

By default, the rule allows access checks can be performed using system Apex provisions such as DescribeSObjectResult.isAccessible/Createable/etc., the SOQL WITH SECURITY\_ENFORCED clause, or using the open source [Force.com ESAPI](https://github.com/forcedotcom/force-dot-com-esapi) class library. Because it is common to use authorization facades to assist with this task, the rule also allows configuration of regular expression-based patterns for the methods used to authorize each type of CRUD operation. These pattern are configured via the following properties:

* createAuthMethodPattern/createAuthMethodTypeParamIndex - a pattern for the method used for create authorization and an optional 0-based index of the parameter passed to that method that denotes the SObjectType being authorized for create.
* readAuthMethodPattern/readAuthMethodTypeParamIndex - a pattern for the method used for read authorization and an optional 0-based index of the parameter passed to that method that denotes the SObjectType being authorized for read.
* updateAuthMethodPattern/updateAuthMethodTypeParamIndex - a pattern for the method used for update authorization and an optional 0-based index of the parameter passed to that method that denotes the SObjectType being authorized for update.
* deleteAuthMethodPattern/deleteAuthMethodTypeParamIndex - a pattern for the method used for delete authorization and an optional 0-based index of the parameter passed to that method that denotes the SObjectType being authorized for delete.
* undeleteAuthMethodPattern/undeleteAuthMethodTypeParamIndex - a pattern for the method used for undelete authorization and an optional 0-based index of the parameter passed to that method that denotes the SObjectType being authorized for undelete.
* mergeAuthMethodPattern/mergeAuthMethodTypeParamIndex - a pattern for the method used for merge authorization and an optional 0-based index of the parameter passed to that method that denotes the SObjectType being authorized for merge.

The following example shows how the rule can be configured for the [sirono-common](https://github.com/SCWells72/sirono-common) [AuthorizationUtil](https://github.com/SCWells72/sirono-common" \l "authorization-utilities) class:

<rule ref="category/apex/security.xml/ApexCRUDViolation" message="Validate CRUD permission before SOQL/DML operation">

<priority>3</priority>

<properties>

<property name="createAuthMethodPattern" value="AuthorizationUtil\.(is|assert)(Createable|Upsertable)"/>

<property name="readAuthMethodPattern" value="AuthorizationUtil\.(is|assert)Accessible"/>

<property name="updateAuthMethodPattern" value="AuthorizationUtil\.(is|assert)(Updateable|Upsertable)"/>

<property name="deleteAuthMethodPattern" value="AuthorizationUtil\.(is|assert)Deletable"/>

<property name="undeleteAuthMethodPattern" value="AuthorizationUtil\.(is|assert)Undeletable"/>

<property name="mergeAuthMethodPattern" value="AuthorizationUtil\.(is|assert)Mergeable"/>

</properties>

</rule>

Note: This rule will produce false positives for VF getter methods. In VF getters the access permission check happens automatically and is not needed explicitly. However, the rule can’t reliably determine whether a getter is a VF getter or not and reports a violation in any case. In such cases, the violation should be [suppressed](https://pmd.github.io/latest/pmd_userdocs_suppressing_warnings.html).

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexCRUDViolationRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexCRUDViolationRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **Contact** **foo(String** status**,** **String** ID**)** **{**

*// validate you can actually query what you intend to retrieve*

**Contact** c **=** **[**SELECT Status\_\_c FROM **Contact** WHERE **Id=:**ID WITH SECURITY\_ENFORCED**];**

*// Make sure we can update the database before even trying*

**if** **(!Schema.**sObjectType**.**Contact**.**fields**.**Status\_\_c**.**isUpdateable**())** **{**

**return** **null;**

**}**

c**.**Status\_\_c **=** status**;**

update c**;**

**return** c**;**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |
| updateAuthMethodPattern |  | A regular expression for one or more custom update authorization method name patterns. | no |
| updateAuthMethodTypeParamIndex | 0 | The 0-based index of the sObjectType parameter for the custom update authorization method. Defaults to 0. | no |
| readAuthMethodPattern |  | A regular expression for one or more custom read authorization method name patterns. | no |
| readAuthMethodTypeParamIndex | 0 | The 0-based index of the sObjectType parameter for the custom read authorization method. Defaults to 0. | no |
| undeleteAuthMethodPattern |  | A regular expression for one or more custom undelete authorization method name patterns. | no |
| undeleteAuthMethodTypeParamIndex | 0 | The 0-based index of the sObjectType parameter for the custom undelete authorization method. Defaults to 0. | no |
| deleteAuthMethodPattern |  | A regular expression for one or more custom delete authorization method name patterns. | no |
| deleteAuthMethodTypeParamIndex | 0 | The 0-based index of the sObjectType parameter for the custom delete authorization method. Defaults to 0. | no |
| mergeAuthMethodPattern |  | A regular expression for one or more custom merge authorization method name patterns. | no |
| mergeAuthMethodTypeParamIndex | 0 | The 0-based index of the sObjectType parameter for the custom merge authorization method. Defaults to 0. | no |
| createAuthMethodPattern |  | A regular expression for one or more custom create authorization method name patterns. | no |
| createAuthMethodTypeParamIndex | 0 | The 0-based index of the sObjectType parameter for the custom create authorization method. Defaults to 0. | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexCRUDViolation" />

**Use this rule and customize it:**

<rule ref="category/apex/security.xml/ApexCRUDViolation">

<properties>

<property name="updateAuthMethodPattern" value="" />

<property name="updateAuthMethodTypeParamIndex" value="0" />

<property name="readAuthMethodPattern" value="" />

<property name="readAuthMethodTypeParamIndex" value="0" />

<property name="undeleteAuthMethodPattern" value="" />

<property name="undeleteAuthMethodTypeParamIndex" value="0" />

<property name="deleteAuthMethodPattern" value="" />

<property name="deleteAuthMethodTypeParamIndex" value="0" />

<property name="mergeAuthMethodPattern" value="" />

<property name="mergeAuthMethodTypeParamIndex" value="0" />

<property name="createAuthMethodPattern" value="" />

<property name="createAuthMethodTypeParamIndex" value="0" />

</properties>

</rule>

ApexCSRF

Deprecated

The rule has been moved to another ruleset. Use instead: [ApexCSRF](https://pmd.github.io/latest/pmd_rules_apex_errorprone.html" \l "apexcsrf)

Deprecated

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Having DML operations in Apex class constructor or initializers can have unexpected side effects: By just accessing a page, the DML statements would be executed and the database would be modified. Just querying the database is permitted.

In addition to constructors and initializers, any method called init is checked as well.

Salesforce Apex already protects against this scenario and raises a runtime exception.

Note: This rule has been moved from category "Security" to "Error Prone" with PMD 6.21.0, since using DML in constructors is not a security problem, but crashes the application.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.errorprone.ApexCSRFRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/errorprone/ApexCSRFRule.java)

**Example(s):**

**public** **class** **Foo** **{**

*// initializer*

**{**

insert data**;**

**}**

*// static initializer*

**static** **{**

insert data**;**

**}**

*// constructor*

**public** **Foo()** **{**

insert data**;**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexCSRF" />

ApexDangerousMethods

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Checks against calling dangerous methods.

For the time being, it reports:

* Against FinancialForce’s Configuration.disableTriggerCRUDSecurity(). Disabling CRUD security opens the door to several attacks and requires manual validation, which is unreliable.
* Calling System.debug passing sensitive data as parameter, which could lead to exposure of private data.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexDangerousMethodsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexDangerousMethodsRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **Foo()** **{**

**Configuration.**disableTriggerCRUDSecurity**();**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexDangerousMethods" />

ApexInsecureEndpoint

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Checks against accessing endpoints under plain **http**. You should always use **https** for security.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexInsecureEndpointRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexInsecureEndpointRule.java)

**Example(s):**

**public** without sharing **class** **Foo** **{**

**void** **foo()** **{**

**HttpRequest** req **=** **new** **HttpRequest();**

req**.**setEndpoint**(**'http:*//localhost:com');*

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexInsecureEndpoint" />

ApexOpenRedirect

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Checks against redirects to user-controlled locations. This prevents attackers from redirecting users to phishing sites.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexOpenRedirectRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexOpenRedirectRule.java)

**Example(s):**

**public** without sharing **class** **Foo** **{**

**String** unsafeLocation **=** **ApexPage.**getCurrentPage**().**getParameters**.**get**(**'url\_param'**);**

**PageReference** **page()** **{**

**return** **new** **PageReference(**unsafeLocation**);**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexOpenRedirect" />

ApexSharingViolations

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Detect classes declared without explicit sharing mode if DML methods are used. This forces the developer to take access restrictions into account before modifying objects.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexSharingViolationsRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexSharingViolationsRule.java)

**Example(s):**

**public** without sharing **class** **Foo** **{**

*// DML operation here*

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexSharingViolations" />

ApexSOQLInjection

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Detects the usage of untrusted / unescaped variables in DML queries.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexSOQLInjectionRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexSOQLInjectionRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **void** **test1(String** t1**)** **{**

**Database.**query**(**'SELECT **Id** FROM **Account**' **+** t1**);**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexSOQLInjection" />

ApexSuggestUsingNamedCred

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Detects hardcoded credentials used in requests to an endpoint.

You should refrain from hardcoding credentials:

* They are hard to mantain by being mixed in application code
* Particularly hard to update them when used from different classes
* Granting a developer access to the codebase means granting knowledge of credentials, keeping a two-level access is not possible.
* Using different credentials for different environments is troublesome and error-prone.

Instead, you should use *Named Credentials* and a callout endpoint.

For more information, you can check [this](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_callouts_named_credentials.htm)

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexSuggestUsingNamedCredRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexSuggestUsingNamedCredRule.java)

**Example(s):**

**public** **class** **Foo** **{**

**public** **void** **foo(String** username**,** **String** password**)** **{**

**Blob** headerValue **=** **Blob.**valueOf**(**username **+** ':' **+** password**);**

**String** authorizationHeader **=** 'BASIC ' **+** **EncodingUtil.**base64Encode**(**headerValue**);**

req**.**setHeader**(**'**Authorization**'**,** authorizationHeader**);**

**}**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexSuggestUsingNamedCred" />

ApexXSSFromEscapeFalse

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Reports on calls to addError with disabled escaping. The message passed to addError will be displayed directly to the user in the UI, making it prime ground for XSS attacks if unescaped.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexXSSFromEscapeFalseRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexXSSFromEscapeFalseRule.java)

**Example(s):**

**public** without sharing **class** **Foo** **{**

**Trigger.**new**[**0**].**addError**(**vulnerableHTMLGoesHere**,** **false);**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 100 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexXSSFromEscapeFalse" />

ApexXSSFromURLParam

**Since:** PMD 5.5.3

**Priority:** Medium (3)

Makes sure that all values obtained from URL parameters are properly escaped / sanitized to avoid XSS attacks.

**This rule is defined by the following Java class:** [net.sourceforge.pmd.lang.apex.rule.security.ApexXSSFromURLParamRule](https://github.com/pmd/pmd/blob/master/pmd-apex/src/main/java/net/sourceforge/pmd/lang/apex/rule/security/ApexXSSFromURLParamRule.java)

**Example(s):**

**public** without sharing **class** **Foo** **{**

**String** unescapedstring **=** **ApexPage.**getCurrentPage**().**getParameters**.**get**(**'url\_param'**);**

**String** usedLater **=** unescapedstring**;**

**}**

**This rule has the following properties:**

| **Name** | **Default Value** | **Description** | **Multivalued** |
| --- | --- | --- | --- |
| cc\_categories | Security | Deprecated Code Climate Categories | yes. Delimiter is ‘|’. |
| cc\_remediation\_points\_multiplier | 50 | Deprecated Code Climate Remediation Points multiplier | no |
| cc\_block\_highlighting | false | Deprecated Code Climate Block Highlighting | no |

**Use this rule with the default properties by just referencing it:**

<rule ref="category/apex/security.xml/ApexXSSFromURLParam" />