PMD Rulesets index: Current Rulesets

List of rulesets and rules contained in each ruleset.

- Android: These rules deal with the Android SDK, mostly related to best practices.To get better results, make sure that the auxclasspath is defined for type resolution to work.
- Basic: The Basic ruleset contains a collection of good practices which should be followed.
- Braces: The Braces ruleset contains rules regarding the use and placement of braces.
- Clone Implementation: The Clone Implementation ruleset contains a collection of rules that find questionable usages of the clone() method.
- Code Size: The Code Size ruleset contains rules that find problems related to code size or complexity.
- Comments: Rules intended to catch errors related to code comments
- Controversial: The Controversial ruleset contains rules that, for whatever reason, are considered controversial.They are held here to allow people to include them as they see fit within their custom rulesets.
- Coupling: Rules which find instances of high or inappropriate coupling between objects and packages.
- Design: The Design ruleset contains rules that flag suboptimal code implementations. Alternate approachesare suggested.
- Empty Code: The Empty Code ruleset contains rules that find empty statements of any kind (empty method,empty block statement, empty try or catch block,…).
- Finalizer: These rules deal with different problems that can occur with finalizers.
- Import Statements: These rules deal with different problems that can occur with import statements.
- J2EE: Rules specific to the use of J2EE implementations.
- Jakarta Commons Logging: The Jakarta Commons Logging ruleset contains a collection of rules that find questionable usages of that framework.
- JavaBeans: The JavaBeans Ruleset catches instances of bean rules not being followed.
- Java Logging: The Java Logging ruleset contains a collection of rules that find questionable usages of the logger.
- JUnit: These rules deal with different problems that can occur with JUnit tests.
- Migration: Contains rules about migrating from one JDK version to another. Don’t use these rules directly,rather, use a wrapper ruleset such as migrating_to_13.xml.
- Naming: The Naming Ruleset contains rules regarding preferred usage of names and identifiers.
- Optimization: These rules deal with different optimizations that generally apply to best practices.
- Security Code Guidelines: These rules check the security guidelines from Sun, published

at http://java.sun.com/security/seccodeguide.html#gcg

- Strict Exceptions: These rules provide some strict guidelines about throwing and catching exceptions.
- String and StringBuffer: These rules deal with different issues that can arise with manipulation of the String, StringBuffer, or StringBuilder instances.
- Type Resolution: These are rules which resolve java Class files for comparison, as opposed to a String
- Unnecessary: The Unnecessary Ruleset contains a collection of rules for unnecessary code.
- Unused Code: The Unused Code ruleset contains rules that find unused or ineffective code.

Android (java)

- CallSuperFirst: Super should be called at the start of the method
- CallSuperLast: Super should be called at the end of the method
- DoNotHardCodeSDCard: Use Environment.getExternalStorageDirectory() instead of "/sdcard"

Basic (java)

- JumbledIncrementer: Avoid jumbled loop incrementers - its usually a mistake, and is confusing even if intentional.
- ForLoopShouldBeWhileLoop: Some for loops can be simplified to while loops, this makes them more concise.
- OverrideBothEqualsAndHashcode: Override both public boolean Object.equals(Object other), and public int Object.hashCode(), or override neither. Even if you are inheriting a hashCode() from a parent class, consider implementing hashCode and explicitly delegating to your superclass.
- DoubleCheckedLocking: Partially created objects can be returned by the Double Checked Locking pattern when used in Java.An optimizing JRE may assign a reference to the baz variable before it creates the object thereference is intended to point to.For more details refer to: http://www.javaworld.com/javaworld/jw-02-2001/jw-0209-double.htmlor http://www.cs.umd.edu/~pugh/java/memoryModel/DoubleCheckedLocking.html
- ReturnFromFinallyBlock: Avoid returning from a finally block, this can discard exceptions.
- UnconditionalIfStatement: Do not use "if" statements whose conditionals are always true or always false.
- BooleanInstantiation: Avoid instantiating Boolean objects; you can reference Boolean.TRUE, Boolean.FALSE, or call Boolean.valueOf() instead.
- CollapsibleIfStatements: Sometimes two consecutive 'if' statements can be consolidated by separating their conditions with a boolean short-circuit operator.
- ClassCastExceptionWithToArray: When deriving an array of a specific class from your Collection, one should provide an array ofthe same class as the parameter of the toArray() method. Doing otherwise you will will resultin a ClassCastException.