John Chau

5/24/2021

SE333

Static Analysis

To begin with, a rule priority of 2 causes PMD static analysis to find “EqualsNull”, “AvoidUsingShortType”, “LongInstantiation”, and “SystemPrintln” is used as problems in the main programs. In the test programs, “LongInstantiation” and “SystemPrintln” are also issues with a rule priority of 2. First, “EqualsNull” is when tests for null use equals() instead of “==” and this should block the build because this actually affects the logic of the program since this is the wrong thing to use to check if an object is null. Therefore, it is listed as high priority. Next, “AvoidUsingShortType” is when the short type is used in Java, and should not block the build unless performance is an issue because it benefits the memory usage at the cost of performance. It is listed as high priority too but I think this is only the case if performance could be an issue. Also, “LongInstantiation” is when you create a Long() object which can be avoided by using the static Long.valueOf() which recycles earlier instances and saves memory and should not block the build unless again, performance is an issue. Its priority is medium high which makes sense. Finally, “System.out.println” is when println’s are used to debug a program and can even remain in production code and clog the Standard out log which shouldn’t block the build because they can be easily removed after debugging and I wouldn’t consider clogging the Standard out log to be a huge issue. Its priority is also medium high which makes sense because it doesn’t really affect the logic of the program.

In the main programs with a rule priority of 3, the problems are “AvoidFieldNameMatchingMethodName”, “MissingOverride”, “InsufficientStringBufferDeclaration”, “AppendCharacterWithChar”, “BeanMembersShouldSerialize”, “EqualsNull”, “AvoidUsingShortType”. “CloseResource”, “LongInstantiation”, “LiteralsFirstInComparison”, and “AvoidLiteralsInIfCondition”. In the test programs with a rule priority of 3, the problems are “JUnitAssertionsShouldIncludeMessage”, “UnusedPrivateMethod”, “AvoidDuplicateLiterals”, and “SystemPrintln”. “AvoidFieldNameMatchingMethodName” is when a field and method name in a class have the same name. “MissingOverride” is when you annotate overridden methods with @Override to ensure you really override one. "InsufficientStringBufferDeclaration” is when a StringBuffer or StringBuilder is incorrectly sized for the correct number of characters. “AppendCharacterWithChar” is when you concatenate characters as strings with the StringBuffer/StringBuilder.append methods. “BeanMembersShouldSerialize” is when a class is a bean, or is referenced by a bean directly or indirectly and needs to be serializable. “CloseResource” is when resources such as java.Sql.Connection are not closed after use. “LiteralsFirstInComparison” is when literals are not placed first in String comparisons. “AvoidLiteralsInIfCondition” is when hard-coded literals are used in conditional statements. “JUnitAssertionsShouldIncludeMessage” is when JUnit assertions do not include a descriptive message. “UnusedPrivateMethod” is self explanatory. “AvoidDuplicateLiterals” is when there are multiple identical literal strings when they can be assigned to a single variable which would make maintainability better.

“EqualsNull” can be fixed by using “==” to check if an object is null instead of equals(). “AvoidUsingShortType” can be fixed by using ints not shorts. “LongInstantiation” can be fixed by using the static Long.valueOf() instead of the Long constructor. “SystemPrintln” can be fixed by using a logger. “LiteralsFirstInComparison” should be fixed by putting literals first in a String comparison. “AvoidLiteralsInIfCondition” should be fixed by using variables instead of literals in if conditions. “AvoidDuplicateLiterals” can be fixed in the same way.