CID 10022 - Missing call to super class

file: ListOrderedMap.java:262

Triage - False Positive

This warning is a false positive as this base class has no need to call its super class. All other six subclass instances out of nine return the key set of the parent class as it is. The ListOrderedMap class returns a new keySetView() instance of the object instead, which is modifiable. So in both cases a KeySet() is returned.

CID 10023 - Missing call to super class

file: FixedSizeSortedMap.java:144

Triage - Intentional

This warning seems to be bad practice. The line the developer of code the developer uses:

—> Set set = map.keySet() is more or less the same as that which is used by the other six subclass instances out of nine —> Set set = super.keySet(). The warning would not appear if the developer used the second line of code instead.

CID 10024 - Missing call to super class

file: FixedSizeMap.java:144

Triage - Intentional

This warning seems to be bad practice. The line the developer of code the developer uses:

—> Set set = map.keySet() is more or less the same as that which is used by the other six subclass instances out of nine —> Set set = super.keySet(). The warning would not appear if the developer used the second line of code instead.

CID 10025 - Dereference after null check

file: ObjectGraphIterator.java:147

Triage - Bug

This is a bug because line 186 of the same file calls a method on this.currentIterator when it might be null, as indicated by the null check on line 140 of the same file. A fix is simply to add a null check for currentIterator before the method call.

CID 10026 - Unguarded read

file: FastArrayList.java:852

Triage - Bug

This is a bug because the function does not get the lock required for sublist, which could result in incorrect values or values modified by another thread being returned. An example fix would be to grab a lock before accessing sublist, following the examples provided by Coverity.

CID 10027 - Dereference null return value

file: DoubleOrderedMap.java:883

Triage - False Positive

Lines 881, 882 check to see whether the left and right children of deletedNode are null or not. Since they are not null and this is a DoubleOrderedMap class, there must be one that is greater and thus nextGreater cannot return null.

CID 10028 - Unguarded write

file: FastArrayList.java:1241

Triage - Bug

This is a bug because the function does not get the lock required for sublist, which could result in incorrect values or values modified by another thread being returned. An example fix would be to grab a lock before accessing sublist, following the examples provided by Coverity.

CID 10029 - Check of thread-shared field evades lock acquisition

file: FastHashMap.java:665

Triage - Bug

If two threads access this data at the same time, there is a chance that the field lastReturned could acquire inconsistent data due to them trying to update it at the same time. A possible fix for this would be update all other threads when one edits the value of lastReturned.

CID 10030 - Thread deadlock

file: FastHashMap.java:548

Triage - Intentional

There is only a single task/line of code that can be executed before another thread could interfere, so there can’t be a deadlock. Intentional because it still might be a better idea to grab all current locks to be certain.

CID 10031 - Dereference null return value

file: TreeList.java:656

Triage - Intentional

While it seems there is a real possibility that getLeftSubTree() could be called upon a null value, we have to notice that rotateLeft() is a private function. Looking through the TreeList class, we can see that the function balance() is the only one that calls rotateLeft() and it does so only after checking that the height is value where ‘faedelung’ does not apply (i.e. a right subtree definitely exists). Still there is always a possibility for human error and a simple null check would be a good idea here (hence not a false positive).

CID 10032 - Check of thread-shared field evades lock acquisition

file: StaticBucketMap.java:513

Triage - Bug

If two threads access this data at the same time, there is a chance that the field bucket could acquire inconsistent data due to them trying to update it at the same time. A possible fix for this would be update all other threads when one edits the value of bucket.

CID 10033 - Arguments in wrong order

file: TreeBidiMap.java:1603

Triage - False Positive

This function call is made for the case that is INVERSEMAPENTRY, in which case the argument switch makes sense.

CID 10034 - Check of thread-shared field evades lock acquisition

file: StaticBucketMap.java:522

Triage - Bug

If two threads access this data at the same time, there is a chance that the field bucket could acquire inconsistent data due to them trying to update it at the same time. A possible fix for this would be update all other threads when one edits the value of bucket.

CID 10035 - Unguarded write

file: FastArrayList.java:1221

Triage - Bug

This is a bug because the function does not get the lock required for sublist, which could result in incorrect values or values modified by another thread being returned. An example fix would be to grab a lock before accessing sublist, following the examples provided by Coverity.

CID 10036 - Check of thread-shared field evades lock acquisition

file: FastTreeMap.java:768

Triage - Bug

If two threads access this data at the same time, there is a chance that the field lastReturned could acquire inconsistent data due to them trying to update it at the same time. A possible fix for this would be update all other threads when one edits the value of lastReturned.

CID 10037 - Dereference null return value

file: TreeList.java:683

Triage - Intentional

While it seems there is a real possibility that getRightSubTree() could be called upon a null value, we have to notice that rotateRight() is a private function. Looking through the TreeList class, we can see that the function balance() is the only one that calls rotateRight() and it does so only after checking that the height is a value where ‘faedelung’ does not apply (i.e. a left subtree definitely exists). Still there is always a possibility for human error and a simple null check would be a good idea here (hence not a false positive).

CID 10038 - Thread Deadlock

file: FastArrayList.java:1136

Triage - Intentional

There is only a single task/line of code that can be executed before another thread could interfere, so there can’t be a deadlock. Intentional because it still might be a better idea to grab all current locks to be certain.

CID 10039 - Dereference null return value

file: TreeBidiMap.java:1019

Triage - False Positive

Line 1018 checks to see whether the left and right children of deletedNode are null or not. Since they are not null and this is a tree class, there must be one that is greater and thus nextGreater cannot return null.

CID 10040 - Thread deadlock

file: FastTreeMap.java:653

Triage - Intentional

There is only a single task/line of code that can be executed before another thread could interfere, so there can’t be a deadlock. Intentional because it still might be a better idea to grab all current locks to be certain.

CID 10041 - Volatile not atomically updated

file: ReferenceMap.java:555

Triage - Bug

Since there is no lock held, it is possible that modCount could be overwritten by an intervening thread and the current thread would then change the value of modCount based on old values that are no longer valid, so this is a bug. Fix would be to grab a lock before modifying the values.

CID 10042 - Volatile not atomically updated

file: ReferenceMap.java:582

Triage - Bug

Since there is no lock held, it is possible that modCount could be overwritten by an intervening thread and the current thread would then change the value of modCount based on old values that are no longer valid, so this is a bug. Fix would be to grab a lock before modifying the values.