

# Distinguishing between the different ProGuard “-keep” directives

If you search for ProGuard rules for a Java or Android library, you’ll see a lot of answers on StackOverflow that tell you to do something like this:

```
-keep class com.foo.library.** { *; }
```

That advice is really bad, and you should never do it. First, it’s overly broad — that double-asterisk in the package means every class under every package under that top-level package; and the asterisk inside the curly braces applies to every member (variables, methods, and constants) inside those class. That is, it applies to all code in the library. If you use that rule, Jake Wharton is going to come yell at you:



JakeWharton commented on Jul 28, 2016





Collaborator



Those rules are crazy. You might as well just turn off ProGuard if you're going to keep 100% of libraries.





Second, and what this post is about, is the beginning of the directive, that “-keep”. *You almost never want to use -keep*; if you do need a ProGuard rule, you usually want one of the more specific variants below. But it always takes me a minute with the ProGuard manual to figure out which one of those variants applies to my case, so I made some tables for quick visual reference. (Quick aside: [the ProGuard manual](#) is very useful and I highly recommend you look through it.)

No rule

|           | classes   | members  |
|-----------|---|--|
| shrink    |  |  |
| obfuscate |  |  |





To get our bearings, let's look at the default. If you don't specify a keep directive of any kind, then ProGuard is going to do its normal thing — it's going to both *shrink* (i.e. remove unused code) and *obfuscate* (i.e. rename things) both classes and class members.

-keep

|           | classes   | members  |
|-----------|---|--|
| shrink    |  |  |
| obfuscate |  |  |





See, this is why I said you should almost never use `-keep`. `-keep` disables all of ProGuard's goodness. No shrinking, no obfuscation; not for classes, not for members. In real use cases, you can let ProGuard do at least *some* of it's work. Even if your variables are accessed by reflection, you could remove and rename unused classes, for example. So let's look through the more specific `-keep` variants.

### `-keepclassmembers`

|           | classes   | members  |
|-----------|---|--|
| shrink    |    |    |
| obfuscate |  |  |





This protects only the members of the class from shrinking and obfuscation. That is, if a class is unused, it will be removed. If the class is used, the class will be kept but renamed. But inside any class that is kept around, all of its members will be there, and they will have their original names.

-keepnames

|           | classes   | members  |
|-----------|---|--|
| shrink    |  |  |
| obfuscate |  |  |

This allows shrinking for classes and members, but not obfuscation. That is, any unused code is going to get removed. But the code that is kept will keep its original names.

-keepclassmembernames

|           | classes   | members  |
|-----------|---|--|
| shrink    |  |  |
| obfuscate |  |  |

This is the most permissive keep directive; it lets ProGuard do *almost* all of its work. Unused classes are removed, the remaining classes are renamed, unused members of those classes are removed, but then the remaining members keep their original names.

### `-keepclasseswithmembers`

This one doesn't get a table, because it's the same as `-keep`. The difference is that it only applies to classes who have all of the members in the class specification.

### `-keepclasseswithmembernames`

Similarly, this rule is the same as `-keepnames`. The difference, again, is that it only applies to classes who have all of the members in the class specification.

## Conclusion

You want to let ProGuard do as much work as possible, so pick the directive that has the fewest red X blocks above, while still meeting your need.



#### **Author: jeb**

Views expressed here are my own and do not necessarily reflect the views of my employer.

[View all posts by jeb](#)



jeb / November 14, 2017 / Android, ProGuard  
jebware.com / Proudly powered by WordPress