ToiletPaper #121

Happily Tweeting Leaks

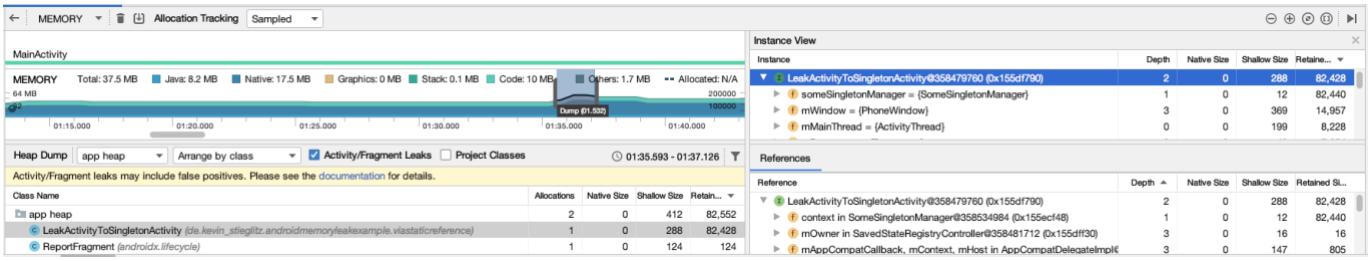
Author: Kevin Stieglitz / Software Engineer / Business Division Automotive World

✕ Problem

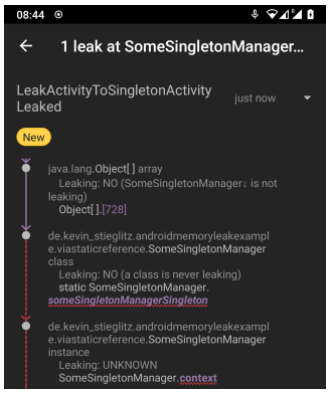
Nobody likes memory leaks. They are usually difficult to reproduce, even harder to identify and, in worst case, after long usage they only become noticeable with an OutOfMemoryException.

✓ Solution

Fortunately, there are several ways to detect leaks in Android. One possibility is an internal offer by Android Studio: with the help of the profiler, the Java heap can be displayed and then analyzed. Since Android Studio 3.6 it can also be filtered specifically for no longer existing but still referenced Activities/Fragments (Activity/Fragment Leaks). Besides the existing instances, the references to them are also displayed, so that conclusions about leaks can be drawn. The screenshot below shows an example application, in which the reference within a singleton is held to one activity, so that the garbage collector cannot clean it up afterwards.



A much more comfortable solution is offered by the library **LeakCanary**. A WeakReference to different instances is created in debug builds (for example, to Activities in the onActivityDestroyed method). A background thread then checks whether the reference has been cleaned up (after the garbage collector has been executed). If not, the heap is saved in an .hprof file. This is analyzed in a separate process using the previous WeakReference and the chain of references is calculated which prevents the object from freeing up its memory. In LeakCanary, possible causes for the memory leak are highlighted with a red wavy line. This makes it quite easy to detect leaks at an early stage and to narrow down their origin efficiently.



+ Further Aspects

- Android app with LeakCanary and multiple popular Leaks: https://github.com/kvn-stgl/AndroidMemoryLeakExample
- LeakCanary: https://square.github.io/leakcanary/