

Search IssueTracker

Sign in

Android Public Tracker > App Development > Android Studio > Deployment > Debugger 288288933

com.sun.jdi.VMDisconnectedException in Debugger Evaluator

+1

Hotlists

Mark as Duplicate

Comments (12)

Dependencies

Duplicates (0)

Blocking (0)

Resources (0)

Assigned

Bug

P2

+ Add Hotlist

STATUS UPDATE

No update yet.

Edit

DESCRIPTION

an...@jetbrains.com created issue #1

STEPS TO REPRODUCE:  
1. Declare an annotation class in your code. For example: annotation class Bar(val test: String)  
2. Run debugger and try to evaluate: Bar::class.primaryConstructor!!.call("s").test  
  
Actual result: com.sun.jdi.VMDisconnectedException and application crash  
Expected result: no error, "s" in evaluator  
  
Studio Build: 2022.1.1 Patch 2  
Kotlin: 1.9.0-Beta  
OS: MacOS Monterey

COMMENTS

ks...@google.com

<ks...@google.com>

Assigned to an...@google.com.

jb...@google.com

<jb...@google.com>

#2

Status: New

reproduced at head.

aa...@google.com

<aa...@google.com>

#3

A few observations:  
1. It works if the class is not an annotation  
2. Bar::class.primaryConstructor!!.call("s") works but accessing the field crashes.  
3. Same code is executed fine in app code, only crashes when evaluated in debugger  
4. Same code works with plain IJ.  
  
Fabien, any insight?  
  
App crashes with SIGSEGV:

|   |                               |  |
|---|-------------------------------|--|
| 2023-06-29 15:26:12.038 28519-28519 libc  | com.alonalbert.kotlinviewsapp | A Fatal signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault |
| 2023-06-29 15:26:12.445 28554-28554 DEBUG | pid-28554                     | A Cmdline: com.alonalbert.kotlinviewsapp                 |
| 2023-06-29 15:26:12.445 28554-28554 DEBUG | pid-28554                     | A pid: 28519, tid: 28519, name: .kotlinviewsapp >>> con  |
| 2023-06-29 15:26:12.446 28554-28554 DEBUG | pid-28554                     | A #14 pc 000000000031aacc /data/data/com.alonalbe        |

aa...@google.com

<aa...@google.com>

#4

Note that you have to add the following dependency:

```
implementation(kotlin("reflect"))
```

aa...@google.com

<aa...@google.com>

#5

anastasia, I'm curious about the usefulness of this. Seems like a very unusual thing to be doing. Can you explain your use case?

aa...@google.com

<aa...@google.com>

#6

Steven, Mads

Any thoughts?

aa...@aoogle.com

<aa...@aoogle.com>

#7

Immediate thoughts are that we should not be able to crash Art with requests from the debugger. We should partner up with the Art team to debug the crash on the Art side and then work back to the debugger. Nicolas, do you know who would be the right person on the Art side to help investigate?

ng...@google.com <ng...@google.com> #8

Adding Mythri. For us to debug further, a tombstone would help.

aa...@google.com <aa...@google.com> #9


Not sure what you mean by `tombstone`. The issue is very easy to reproduce.

ng...@google.com <ng...@google.com> #10

A bugreport can do.

aa...@google.com <aa...@google.com> #11

attached

 **br.zip**  
8.5 MB [Download](#)

ng...@google.com <ng...@google.com> #12

Assigned to vm...@google.com.

Crash in check JNI. Vladimir, could you take a look?

```

*** ** Build fingerprint: 'google/sdk_gphone64_x86_64/emu64xa:13/TPB3.220513.017/8678579:userdebug/dev-keys'
Revision: '0'
ABI: 'x86_64'
Timestamp: 2023-06-30 07:15:31.290171996-0700
Process uptime: 17s
Cmdline: com.alonalbert.kotlinviewsapp
pid: 23819, tid: 23819, name: .kotlinviewsapp >>> com.alonalbert.kotlinviewsapp <<<
uid: 10168
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x0000000000000000
Cause: null pointer dereference
    rax 0000000000000001  rbx 00007ffd620210a0  rcx 0000000000000000  rdx 0000000000000000
    r8  0000000000000000  r9  0000000000001a5f  r10 00007225753ad9d0  r11 0000000000000001
    r12 0000000000000002  r13 000072249f518be3  r14 00007ffd62021270  r15 00007ffd62021280
    rdi 00007ffd620210a0  rsi 0000000000000004c
    rbp 000000000000004c  rsp 00007ffd62021020  rip 000072249fbe9597

backtrace:
#00 pc 000000000005e957 /apex/com.android.art/lib64/libart.so (art::(anonymous namespace)::VarArgs::GetValue(char)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#01 pc 000000000005e8bc /apex/com.android.art/lib64/libart.so (art::(anonymous namespace)::ScopedCheck::CheckPossibleHeapValue(art::ScopedObjectAccess&, bool, char)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#02 pc 000000000005e7e16 /apex/com.android.art/lib64/libart.so (art::(anonymous namespace)::ScopedCheck::Check(art::ScopedObjectAccess&, bool, char)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#03 pc 000000000005ed84d /apex/com.android.art/lib64/libart.so (art::(anonymous namespace)::CheckJNI::CheckCallArgs(art::ScopedObjectAccess&, art::ScopedObjectAccess&, bool, char)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#04 pc 000000000005ee3af /apex/com.android.art/lib64/libart.so (art::(anonymous namespace)::CheckJNI::CallMethodA(char const*, _JNIEnv*, _jobject*, _jclass*, bool)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#05 pc 000000000005db1cf /apex/com.android.art/lib64/libart.so (art::(anonymous namespace)::CheckJNI::CallNonvirtualObjectMethodA(_JNIEnv*, _jobject*, _jclass*, int, bool)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#06 pc 0000000000034ff1 /apex/com.android.art/lib64/libjdw.so (invoker_doInvoke+1551) (BuildId: 55ec45ac540f2746bb102c6eb83c002d)
#07 pc 0000000000031b7f /apex/com.android.art/lib64/libjdw.so (event_callback+1633) (BuildId: 55ec45ac540f2746bb102c6eb83c002d)
#08 pc 000000000002e138 /apex/com.android.art/lib64/libjdw.so (cbBreakpoint+392) (BuildId: 55ec45ac540f2746bb102c6eb83c002d)
#09 pc 000000000004d80a /apex/com.android.art/lib64/libopenjdkjvmti.so (openjdkjvmti::JvmtiMethodTraceListener::DexPcMoved(art::Thread*, art::Handle<art::DexCache>)+119) (BuildId: 55ec45ac540f2746bb102c6eb83c002d)
#10 pc 0000000000056c573 /apex/com.android.art/lib64/libart.so (art::instrumentation::Instrumentation::DexPcMovedEventImpl(art::Thread*, art::ObjectPtr<art::DexCache>)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#11 pc 000000000003960cb /apex/com.android.art/lib64/libart.so (art::interpreter::InstructionHandler<false, false, art::Instruction::Format26>::DoExecute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFrame const&)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#12 pc 0000000000038af8a /apex/com.android.art/lib64/libart.so (void art::interpreter::ExecuteSwitchImplCpp<false, false>(art::interpreter::SwitchInstruction const&, art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFrame const&)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#13 pc 0000000000037f3b5 /apex/com.android.art/lib64/libart.so (ExecuteSwitchImplAsm+5) (BuildId: b263fa077d3022123c1495737bc0762f)
#14 pc 00000000000326290 /data/data/com.alonalbert.kotlinviewsapp/code_cache/.overlay/base.apk/classes.dex (com.alonalbert.kotlinviewsapp.MainActivity.onClick+140) (BuildId: b263fa077d3022123c1495737bc0762f)
#15 pc 000000000005724d7 /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFrame const&)+119) (BuildId: b263fa077d3022123c1495737bc0762f)
#16 pc 00000000000911e3d /apex/com.android.art/lib64/libart.so (artQuickToInterpreterBridge+1021) (BuildId: b263fa077d3022123c1495737bc0762f)
#17 pc 0000000000037d49c /apex/com.android.art/lib64/libart.so (art_quick_to_interpreter_bridge+140) (BuildId: b263fa077d3022123c1495737bc0762f)
#18 pc 00000000000037f9f /apex/com.android.art/lib64/libart.so (BuildId: b263fa077d3022123c1495737bc0762f)

```