<clinit> sometimes will not be invoked when calling static methods at first. #12016

Closed **ZekerZhayard** opened this issue on Feb 21, 2021 · 5 comments · Fixed by #12148

Assignees

Labels comp:vm userRaised

Milestone Release 0.26 (Jav...

ZekerZhayard commented on Feb 21, 2021 • edited

```
openjdk version "1.8.0_282"
OpenJDK Runtime Environment (build 1.8.0_282-b08)
Eclipse OpenJ9 VM (build openj9-0.24.0, JRE 1.8.0 Windows 8.1 amd64-64-Bit 20210120_560 (JIT enabled, AOT enabled)
OpenJ9   - 345e1b09e
OMR      - 741e94ea8
JCL      - ab07c6a8fd based on jdk8u282-b08)
```

Considering test cases below:

- Test Case 1

```java
import java.util.Arrays;
import java.util.function.Consumer;

import sun.misc.SharedSecrets;
import sun.reflect.ConstantPool;

public class Main {
    public static void main(String[] args) {
        Consumer<Object> consumer = Test::test;
        ConstantPool cp = SharedSecrets.getJavaLangAccess().getConstantPool(consumer.getClass());
        for (int i = cp.getSize() - 1; i >= 0; i--) {
            try {
                System.out.println("index: " + i + ", method: " + cp.getMethodAtIfLoaded(i) + ", info: " + Arrays.toString(cp.getMemberRefInfoAt(i)));
            } catch (Throwable ignored) {

            }
        }
        Test.test(null);
        System.out.println(Test.a);
    }

    public static class Test {
        public static boolean a = false;

        static {
            System.out.println("<clinit> was invoked");
            throwException();
        }

        public static void throwException() {
            throw new RuntimeException();
        }

        public static void test(Object o) {
            System.out.println("test was invoked");
        }
    }
}
```

In this case, `Test` class will throw an exception in `<clinit>`, then run with OpenJ9 and Hotspot respectively. The result is:

```
--- OpenJ9 ---
index: 5, method: null, info: [Main$Test, test, (Ljava/lang/Object;)V]
index: 2, method: public java.lang.Object(), info: [java/lang/Object, <init>, ()V]
test was invoked
<clinit> was invoked
Exception in thread "main" java.lang.ExceptionInInitializerError
        at java.lang.J9VMInternals.ensureError(J9VMInternals.java:146)
        at java.lang.J9VMInternals.recordInitializationFailure(J9VMInternals.java:135)
        at Main.main(Main.java:19)
Caused by: java.lang.RuntimeException
        at Main$Test.throwException(Main.java:31)
        at Main$Test.<clinit>(Main.java:27)
        ... 1 more

--- Hotspot ---
index: 18, method: public static void Main$Test.test(java.lang.Object), info: [Main$Test, test, (Ljava/lang/Object;)V]
index: 10, method: public java.lang.Object(), info: [java/lang/Object, <init>, ()V]
<clinit> was invoked
Exception in thread "main" java.lang.ExceptionInInitializerError
        at Main.main(Main.java:18)
Caused by: java.lang.RuntimeException
        at Main$Test.throwException(Main.java:31)
        at Main$Test.<clinit>(Main.java:27)
        ... 1 more
```

We will find that, with OpenJ9, `<clinit>` won't be invoked when `Test.test` being invoked and until `Test.a` was acquired.

We can also find that OpenJ9 can't get the method by `cp.getMethodAtIfLoaded(i)` but Hotspot can.

- **Test Case 2**

If we change `cp.getMethodAtIfLoaded(i)` to `cp.getMethodAt(i)`, there will be more different behaviors between OpenJ9 and Hotspot

```
import java.util.Arrays;
import java.util.function.Consumer;

import sun.misc.SharedSecrets;
import sun.reflect.ConstantPool;

public class Main {
    public static void main(String[] args) {
        Consumer<Object> consumer = Test::test;
        ConstantPool cp = SharedSecrets.getJavaLangAccess().getConstantPool(consumer.getClass());
        for (int i = cp.getSize() - 1; i >= 0; i--) {
            try {
                System.out.println("index: " + i + ", method: " + cp.getMethodAt(i) + ", info: " + Arrays.toString(cp.getMemberRefInfoAt(i)));
            } catch (Throwable ignored) {

            }
        }
        Test.test(null);
        System.out.println(Test.a);
    }

    public static class Test {
        public static boolean a = false;

        static {
            System.out.println("<clinit> was invoked");
            throwException();
        }

        public static void throwException() {
            throw new RuntimeException();
        }

        public static void test(Object o) {
            System.out.println("test was invoked");
        }
    }
}
```

Result:

```
--- OpenJ9 ---
<clinit> was invoked
index: 5, method: null, info: [Main$Test, test, (Ljava/lang/Object;)V]
index: 2, method: public java.lang.Object(), info: [java/lang/Object, <init>, ()V]
test was invoked
Exception in thread "main" java.lang.NoClassDefFoundError: Main$Test (initialization failure)
        at java.lang.J9VMInternals.initializationAlreadyFailed(J9VMInternals.java:96)
        at Main.main(Main.java:19)
Caused by: java.lang.RuntimeException
        at Main$Test.throwException(Main.java:31)
        at Main$Test.<clinit>(Main.java:27)
        at sun.reflect.ConstantPool.getMethodAt0(Native Method)
        at sun.reflect.ConstantPool.getMethodAt(ConstantPool.java:41)
        at Main.main(Main.java:13)

--- Hotspot ---
index: 18, method: public static void Main$Test.test(java.lang.Object), info: [Main$Test, test, (Ljava/lang/Object;)V]
index: 10, method: public java.lang.Object(), info: [java/lang/Object, <init>, ()V]
<clinit> was invoked
Exception in thread "main" java.lang.ExceptionInInitializerError
        at Main.main(Main.java:18)
Caused by: java.lang.RuntimeException
        at Main$Test.throwException(Main.java:31)
        at Main$Test.<clinit>(Main.java:27)
        ... 1 more
```

We will find that `<clinit>` was invoked at `cp.getMethodAt(i)` with OpenJ9 and still unable to get the `Test.test` method.

- **Test Case 3**

If `Test` class no longer throws an exception:

```
import java.util.Arrays;
import java.util.function.Consumer;

import sun.misc.SharedSecrets;
import sun.reflect.ConstantPool;

public class Main {
    public static void main(String[] args) {
        Consumer<Object> consumer = Test::test;
        ConstantPool cp = SharedSecrets.getJavaLangAccess().getConstantPool(consumer.getClass());
        for (int i = cp.getSize() - 1; i >= 0; i--) {
            try {
                System.out.println("index: " + i + ", method: " + cp.getMethodAt(i) + ", info: " + Arrays.toString(cp.getMemberRefInfoAt(i)));
            } catch (Throwable ignored) {

            }
        }
        Test.test(null);
        System.out.println(Test.a);
    }

    public static class Test {
        public static boolean a = false;

        public static void test(Object o) {
            System.out.println("test was invoked");
        }
    }
}
```

Result:

```
--- OpenJ9 ---
<clinit> was invoked
index: 5, method: public static void Main$Test.test(java.lang.Object), info: [Main$Test, test, (Ljava/lang/Object;)V]
index: 2, method: public java.lang.Object(), info: [java/lang/Object, <init>, ()V]
test was invoked
false

--- Hotspot ---
index: 18, method: public static void Main$Test.test(java.lang.Object), info: [Main$Test, test, (Ljava/lang/Object;)V]
index: 10, method: public java.lang.Object(), info: [java/lang/Object, <init>, ()V]
<clinit> was invoked
test was invoked
false
```

In this case, OpenJ9 can get `test` method correctly, but `<clinit>` was still invoked earlier than Hotspot.

👍 2

ZekerZhayard changed the title ~~**<clinit> sometimes will be invoked after static method being invoked.**~~ **<clinit> sometimes will be invoked after static methods.** on Feb 21, 2021

ZekerZhayard changed the title ~~**<clinit> sometimes will be invoked after static methods.**~~ **<clinit> sometimes will not be invoked when calling static methods.** on Feb 21, 2021

ZekerZhayard changed the title ~~**<clinit> sometimes will not be invoked when calling static methods.**~~ **<clinit> sometimes will not be invoked when calling static methods at first.** on Feb 21, 2021

pshipton added the **userRaised** label on Feb 22, 2021

pshipton added this to the **Release 0.26 (Java 8, 11, 16) Apr refresh** milestone on Feb 22, 2021

pshipton commented on Feb 22, 2021 — Contributor

@tajila fyi

ZekerZhayard mentioned this issue on Feb 22, 2021

Avoid initializing class when resolving lambda type. MinecraftForge/typetools#3

Closed

tajila self-assigned this on Feb 23, 2021

pshipton added the comp:vm label on Feb 23, 2021

ZekerZhayard closed this as completed on Mar 2, 2021

ZekerZhayard reopened this on Mar 2, 2021

gacholio commented on Mar 4, 2021 • edited — Contributor

The issue in test 1 appears to be that the ConstantPool code is resolving the static method without running the `<clinit>`. This fails because the `<clinit>` check for `invokestatic` is done in the resolve called from `initialStaticMethod`, which does not run if the method has been resolved.

gacholio commented on Mar 4, 2021 — Contributor

Nope, that's not it - it appears to be somehow caused by the `Test::test` - removing that makes it work as expected.

gacholio commented on Mar 4, 2021 — Contributor

The entry is resolved by this:

```
#0  resolveStaticMethodRef (vmStruct=0x16700, ramCP=0x130ab0, cpIndex=2, resolveFlags=1024)
    at resolvesupport.cpp:649
#1  0x00007ffff5bdc15b in resolveMethodHandleRefInto (vmThread=0x16700, ramCP=0x130ab0, cpIndex=10,
    resolveFlags=1024, ramCPEntry=0x130b50) at resolvesupport.cpp:2005
#2  0x00007ffff5bdc435 in resolveMethodHandleRef (vmThread=0x16700, ramCP=0x130ab0, cpIndex=10, resolveFlags=1024)
    at resolvesupport.cpp:2077
#3  0x00007fffef18c760 in Java_java_lang_invoke_MethodHandleResolver_getCPMethodHandleAt (env=0x16700,
    unusedClass=0x65a30, constantPoolOop=0xbe660, cpIndex=10) at common/sun_reflect_ConstantPool.c:758
```

https://github.com/eclipse/openj9/blob/d780e8d8b86e7c62a1f80411d71bec436574f0f2/runtime/jcl/common/sun_reflect_ConstantPool.c#L755-L758

gacholio assigned **gacholio** and unassigned **tajila** on Mar 4, 2021

gacholio commented on Mar 5, 2021 — Contributor

I have all of the testcases working as expected, but the code needs some reworking before it can be committed.

👍 1 🎉 1

gacholio added a commit to gacholio/openj9 that referenced this issue on Mar 8, 2021

`Correctly load/initialize classes when using ConstantPool` ··· `3130a03`

gacholio mentioned this issue on Mar 8, 2021

Correctly load/initialize classes when using ConstantPool #12148

Merged

gacholio added a commit to gacholio/openj9 that referenced this issue on Mar 8, 2021

`Correctly load/initialize classes when using ConstantPool` ··· `840f3af`

tajila mentioned this issue on Mar 9, 2021

Add ConstantPool reflection tests #12157

Closed

tajila closed this as completed in #12148 on Mar 9, 2021

---

gacholio added a commit to gacholio/openj9 that referenced this issue on Mar 9, 2021

`Correctly load/initialize classes when using ConstantPool` ··· `0c0b7db`

gacholio mentioned this issue on Mar 9, 2021

(0.26) Correctly load/initialize classes when using ConstantPool #12164

Merged

Assignees

gacholio

Labels

comp:vm **userRaised**

Projects

None yet

Milestone

Release 0.26 (Java 8, 11, 16) Apr ref...

Development

Successfully merging a pull request may close this issue.

Correctly load/initialize classes when using ConstantPool
gacholio/openj9

4 participants