链滴

java 类的主动引用和被动引用

作者：fyzzz
原文链接：https://ld246.com/article/1562297765752
来源网站：链滴

本文参考《深入理解JAVA虚拟机》第2版，此书JDK版本为1.7。

主动引用

java类的初始化阶段，虚拟机规范严格规定了5种情况必须立即对类进行初始化。

1. 遇到new、getstatic、putstatic或invokestatic这4条字节码指令时，如果类没有进行过初始化，需要先触发其初始化。

2. 使用java.lang.reflect包的方法对类进行反射调用的时候，如果类没有进行过初始化，则需要先触其初始化。

3. 当初始化一个类时，如果发现其父类没有进行过初始化，则需要先触发其父类的初始化。

4. 当虚拟机启动时，用户需要制定一个要执行的主类（包含main()方法的那个类），虚拟机会先初始这个类。

5. 当使用JDK1.7的动态语言支持时，如果一个java.lang.invoke.MethodHandle实例最后的解析结果EF_geStatic、REF_putStatic、REF_invokeStatic的方法句柄，并且这个方法句柄所对应的类没有进过初始化，则需要先触发其初始化。

被动引用

除了上述5种场景，其他所有类的方式都不会触发初始化，称为被动引用。下面举3个例子来说明。

1、子类引用父类静态变量

```
public class SuperClass{

    static {
        System.out.println("SuperClass init!");
    }

    public static int value = 123;

}

public class SubClass extends SuperClass{

    static {
        System.out.println("SubClass init!");
    }
}

public class InitTest{

    public static void main(String[] args){
        System.out.println(SubClass.value);
    }

}
//执行结果
//SuperClass init!
//123
```

结论：对于静态变量，只有对应这个静态变量的类才会被初始化，通过子类调用只会使父类初始化，类不会初始化。

2、创建对象数组

```
public class InitTest{

    public static void main(String[] args){
        SuperClass[] array = new SuperClass[10];
    }

}
//运行后没有输出
```

结论：创建对象数组不会使对象的类初始化。注：会使数组类初始化，在这个例子中，数组类是"[L包.SuperClass"

3、调用类中的常量

```
public class ConstClass{

    static {
        System.out.println("ConstClass init!");
    }

    public static int value = 123;

}

public class InitTest{

    public static void main(String[] args){
        System.out.println(ConstClass.value);
    }

}
//执行结果
//123
```

结论：调用对象常量不会触发类初始化。书中解释：在编译阶段通过常量传播优化，已经将此常量的“123”存储到了InitTest类的常量池中，以后InitTest对常量ConstClass.value的引用实际都转换为InitTest对自身常量池的引用。