UNIVERSITÄT
BERN

Where does this null come from ?

Bachelor Thesis

Lina Tran

from

Biel/Bienne BE, Switzerland

Faculty of Science
University of Bern

31. July 2016

Prof. Dr. Oscar Nierstrasz

Research assistant Nevena Milojković
Research assistant Boris Spasojević

Software Composition Group

Institute for Computer Science
University of Bern, Switzerland

Abstract

A previous study found out that NullPointerExceptions are very serious in Java projects. When a NullPointerExceptions occurs the developer is provided only with a stack trace to where the exception happened. This only gives insight into the effect of the fault but not into its cause. So we have to ask the question when and why this reference was set to null.

The aim of the project is to be able to provide the user with an additional stack trace of where the value was actually set to null, next to the normal stack trace of an exception. We attempt to achieve this goal by instrumenting java source code ideally with a minimal overhead.

By tracking the null assignments the debugging after a NullPointerException will be simplified.

Contents

1 Introduction **1**

2 Technical Background **3**
2.1 Javassist 3
2.2 JAD 6

3 NullSpy **7**
3.1 High level overview 7
3.2 Low level overview 7
3.3 Challanges 7
3.4 Limitations 8

4 Validation **9**
4.1 JHotDraw 9

5 Conclusion and Future Work **10**

6 Anleitung zu wissenschaftlichen Arbeiten **11**

1 Introduction

Nowadays, certainly every programmer is confronted with NullPointerExceptions in big Java Projects, whether it is for an enterprise or for private purposes. Not to mention even in small Java Projects they are also heavily present.

So what are those NullPointerExceptions? This thesis is going to attach importance to Java that is a concurrent, class-based, object-oriented programming language. We chose Java because NullPointerExceptions are more serious in this language than in others, e.g. Smalltalk. NullPointerException is a RuntimeException. In Java, an object reference can be assigned with a special null value. The exception is thrown when an application attempts to use an object reference that has the null value. (There are multiple ways this exception can be thrown, like: Calling an instance method on the object referred by a null reference; Accessing or modifying an instance field of the object referred by a null reference and so on.) In Java Projects developers always have to deal with a huge amount of references which means avoiding these NullPointerExceptions is as good as impossible.

On regular meetings among programmers they report what they have been doing and what they are planning to do for the next few weeks. But all too often it is stated that they are trying to fix bugs or have spent a lot of time fixing them. If there would be a way to minimize the time fixing exceptions and allow to work more efficiently, projects would progress much faster.

The main goal of the NullSpy application takes a step to that ideal vision. Anytime

developers are facing a NullPointerException they dont have to spent time on debugging finding where and why a reference was set to null. With NullSpy the exact location of the null assignment is shown next to the ordinary stack trace the Java virtual machine produces.

In this thesis it is explained how the goal mentioned above is achieved step by step, by using a class library Javassist (Java Programming Assistant) which allows us to deal with Java bytecode.

2

Technical Background

This chapter provides a short overview of works/technologies used in this project.

2.1 Javassist

Javassist or Java Programming Assistant[1], a subproject of Jboss, is a class library which allows you to deal with Java bytecode. Since 1999 it is used as an engineering toolkit in a broad domain, and is still being extended by Shigeru Chiba. It enables developers to manipulate Java bytecode in a simplified way like defining a new class at runtime or modifying a class file when it is loaded by the JVM. All manipulations are performed at load-time through a provided class loader.

[1] `http://jboss-javassist.github.io/javassist/`

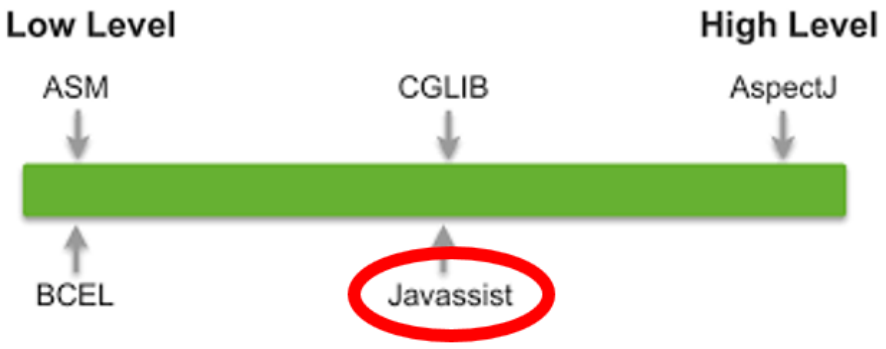

Figure 2.1: Bytecode modification levels

Unlike many other libraries Javassist offers two levels of API: source level and bytecode level (See figure 2.1). Using the source-level API, the user can edit a class file without any familiarity with the specifications of the Java bytecode. Only knowing the Java language is enough because the API is designed only with the vocabulary of Java. On this level the programmer just has to write normal source code and Javassist compiles it automatically. The bytecode level allows the user to modify classes directly in binary form like other editors, e.g. ASM.

At this point, let us look at a small example to give you an idea how the bytecode manipulation works.

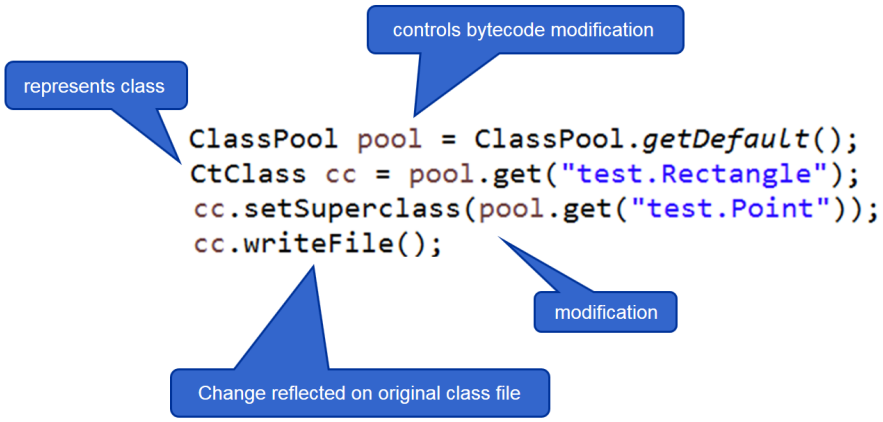

Figure 2.2: Javassist example

First a ClassPool object is obtained that controls bytecode modification with Javassist. With the ClassPool a class file can be read on demand for constructing a CtClass object.

The class CtClass (compile-time class) is just an abstract representation of a class file which means all manipulations are performed on the CtClass. With the method invocation get() on ClassPoola reference to the class file test.Rectangle is obtained. In this example the superclass of test.Rectangle is just changed to test.Point. If the changes are done, the method call writeFile() on CtClass is necessary to make sure that the changes are reflected on the original class file.

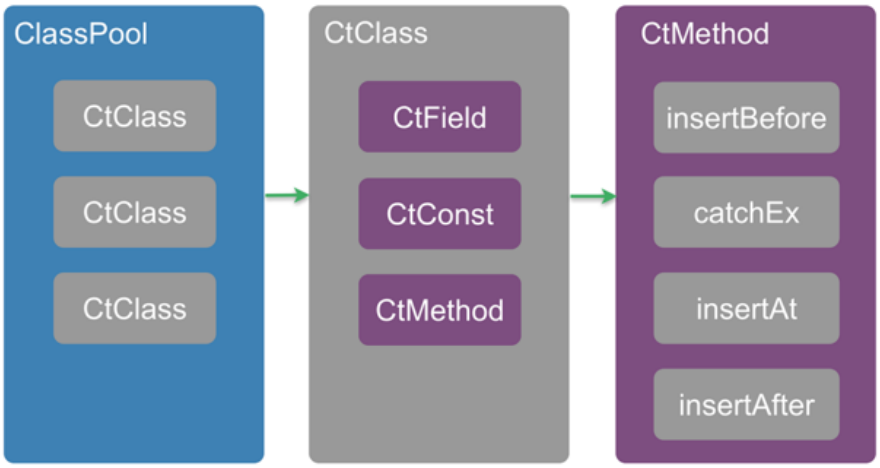

Figure 2.3: Javassist Modules

Figure 2.3 gives you an understanding/overview how the main part of bytecode manipulation with Javassist is built up. The ClassPool is nothing else than a container of multiple CtClasses. As described before CtClass is just the abstract representation of a class file on which modifications are done. Like typical classes, it can hold several compile-time fields, constants or methods. While speaking about bytecode manipulation all the time, nothing but editing method is manly meant. It is possible to insert source code at the beginning of the method body, at the end or at a specific line. Next to these options a catchBlock can even be added.

```
method.insertAt(8,
        "runtimeSupporter.NullDisplayer.test( \""
                + method.getDeclaringClass().getName() + "\","
                + variableName + ");");
```

```
ch.unibe.scg.nullSpy.runtimeSupporter.NullDisplayer.test(
        "className", variable);
```

Figure 2.4: Inserting code example

2.2 JAD

JAva Decompiler[2] is a decompiler and a Eclipe plugin for the programming language Java. A short explanation what a decompiler is: a computer program that takes an executable file as input, and attempts to create a high level, compatible source file that does the same thing. So it is used in software reverse engineering.

JAD is used in NullSpy since after running NullSpy on a project only the modificated bytecodes are available. But to simplify the check if the modification by Javassist, e.g. inserting source code, has succeeded, a decompiler is needed.

[2] `https://sourceforge.net/projects/jadclipse/`

3
NullSpy

In which we understand what the problem is in detail.

3.1 High level overview

blb...

3.2 Low level overview

...

3.3 Challanges

...

3.4 Limitations

...

4
Validation

In which you show how well the solution works.

4.1 JHotDraw

JHotDraw To check whether the logic of the bytecode manipulation in this project NullSpy is working as desired, we had to perform them on a large working project. Thanks to Nevena Milojkovic and her experience with the combination Javassist and JHotDraw we as well decided to test NullSpy on the project JHotDraw. It is an open-source Java GUI framework for technical and structured Graphics. Its original authors have been Erich Gamma and Thomas Eggenschwiler.

5

Conclusion and Future Work

In which we step back, have a critical look at the entire work, then conclude, and learn what lays beyond this thesis.

6

Anleitung zu wissenschaftlichen Arbeiten

This consists of additional documentation, e.g. a tutorial, user guide etc. Required by the Informatik regulation.

Bibliography