Inhalt:

1. Motivation
2. Codeanalyse
   1. Statisch
   2. Dynamisch
   3. Vergleich
3. Compilerpatterns
   1. Insbesondere Visitor
4. Java Bytecode
   1. Opcodes
   2. Stack
5. Erweiterte Motivation
   1. Code Reusability
6. Technologien
   1. Spotbugs
7. Praktische Umsetzung
8. Schluss
9. Literaturverzeichnis

# 1. Motivation

In vielen Projekten kommt es heutzutage zu unzähligen Wiederholungen von Code. Dieser Umstand ist nicht nur kurzfristig von Nachteil, da für jede neue Iteration desselben Codes ein Entwickler sich die Zeit nehmen muss diesen nach bestem Wissen und Gewissen zu implementieren, sondern auch langfristig von Nachteil, da man bei jeder Änderung des Codes auch daran denken muss die anderen Codewiederholungen zu überarbeiten.

Ein erster Schritt hierfür ist es sich wiederholende Codefragmente in Methoden auszulagern und diese zu verwenden. Hierbei kann es jedoch passieren, dass jedes der vorliegenden Codefragmente Kleinigkeiten anders macht. Diese Unterschiede sollten möglichst verringert werden, damit die jeweilige Methode an möglichst vielen (bestenfalls allen) Stellen verwendet werden kann.

Des Weitern kann sich das Problem ergeben, dass der Methodenkopf, der ausgelagerten Methode, nicht an jede Stelle passt, da beispielsweise an der einen Stelle eine „ArrayList“ und an der anderen Stelle eine „LinkedList“ verwendet wird und diese Typen beispielsweise in der Programmiersprache Java nicht implizit aufeinander gecastet werden können.

Dieses Problem soll von dieser Arbeit aufgegriffen und gelöst werden. Das Ziel soll somit sein, dass ein Programm geschrieben wird, welches den Sourcecode nach Methoden durchsucht, welche beispielsweise als Übergapeparameter eine „ArrayList“ hat, jedoch keine der Methoden benutzen, welche nicht auch schon durch eine allgemeinere Klasse oder ein Interface, wie zum Beispiel „List“, welches sowohl von einer „ArrayList“ als auch von einer „LinkedList“ implementiert wird, implementiert werden.

Als Hilfsmittel für die Codeanalyse wird „Spotbugs“ verwendet. „Spotbugs“ ist ein Programm, um Fehler zu finden, wobei während der Compile-Time bereits klar ist, dass Fehler beziehungsweise allgemeine Ungereimtheiten vorhanden sind, jedoch selbst keinen Compile-Fehler erzeugen. Des Weiteren kann man für Spotbugs eigene Plugins entwickeln, welches spezifische Fehler entdeckt. Das Endprodukt dieser Arbeit soll somit ein fertig entwickeltes Spotbugs-Plugin sein, dass Fehler, wie im Absatz vorher beschrieben entdeckt. Weiter soll dieses Plugin Fehler der Art: {WEITERE FEHLEARTEN EINFÜGEN}

{EBENSO DAS VORGEHEN INNERHALB DER ARBEIT ERLÄUTERN}