

# FindBugs & PMD

## Java-Testwerkzeuge

Martin Schwarick

**Pmd**  
DON'T SHOOT THE MESSENGER

# PMD

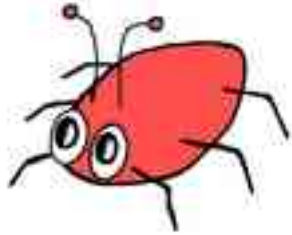
## Steckbrief

- Analysewerkzeug für Java Quellcode ( Open Source)
- Findet u.a.:
  - unbenutzte lokale Variablen
  - leere catch-Blöcke und if-Statements
  - unbenutzte Parameter
  - unbenutzte private Methoden
  - doppelte Importe
  - zu lange/kurze Variablen/Methodennamen

# PMD

## Steckbrief

- Benutzung über Kommandozeile
- zu überprüfende Kriterien werden als Rule Sets ( XML) formuliert ↙ können auch selbst definiert werden
- Resultate werden in XML oder HTML ausgegeben
- zusätzl. Tool zum Finden kopierter Codeteile
- <http://pmd.sourceforge.net>
  - Download: bin oder src



# FindBugs

# FindBugs

## Steckbrief

statischer Bug Pattern Detektor für Java

- Untersuchung der Class-Dateien mit Hilfe von BCEL
- entwickelt von William Puth und David Hovemeyer an der Universität von Maryland

# FindBugs

## Steckbrief

- Open Source Projekt
- <http://findbugs.sourceforge.net>
  - Download: bin oder src
- aktuellste Version: findbugs-0.7.3
- GUI oder Kommandozeile

# FindBugs

## Was wird gemacht?

- 45 implementierte Bug Pattern Detektoren
- Nutzung von BCEL und dem Visitor-Pattern
- Implementierung möglichst einfacher Techniken
- Kategorien:
  - Klassen-Struktur und Vererbungshierarchien
  - Linearer Code-Scan
  - Steuerfluss-orientiert
  - Datenfluss-orientiert

# FindBugs

## BCEL

- Byte Code Engineering Library
- bietet Möglichkeit zur Analyse, Erzeugung und Manipulation von Class-Dateien (Byte Code)
- Infos: <http://jakarta.apache.org/bcel/>

FindBugs

# Bug Pattern

Code-Konstruktionen mit einer hohen  
Fehlerwahrscheinlichkeit

# FindBugs

## Bug Pattern

### „covariante“ Definition von Equals()

– Bsp.:

```
public class Foo extends Object{  
    public boolean equals(Foo obj){ ...}  
    ....  
}
```

Detektion: Untersuchung Methodensignaturen einer Klasse und ihrer Superklassen

# FindBugs

## Bug Pattern

### Equals() und hashCode()

– gleiche Objekte müssen beide Methoden redefinieren  
sonst: kann es möglich sein, dass zwei gleiche Objekte unterschiedliche Hash-werte haben

⚡ Verletzung der Semantik von Hash-tabellen

Detektion: Untersuchung der Methodensignaturen und der Klassenhierarchie

# FindBugs

## Bug Pattern

### Null Pointer Dereferenzierung

- Untersuchung von Instruktionen nur innerhalb von Methoden

- Bsp.:

```
if (foo == null){ // hier drin ist foo null
```

```
    ...
```

```
}
```

Detektion: Datenflussanalyse

# Beispiel

```
// Eclipse 2.1.0,  
// org.eclipse.jdt.  
// internal.ui.javaeditor,  
// ClassFileEditor.java, line 225  
  
if (entry == null) {  
    IClasspathContainer container=  
        JavaCore.getClasspathContainer(  
            entry.getPath(),                // entry is null!  
            root.getJavaProject());  
    ....}
```

# FindBugs

## Bug Pattern

### Offene Streams

- fehlendes Schließen von Input- oder Output Streams
- werden zwar bei der GC geschlossen, aber wann?
- Pufferdaten werden möglicherweise nie geschrieben

Detektion: Suche nach Streamobjekten, die nicht auf allen Pfaden geschlossen werden. ↙ Datenflussanalyse

# Beispiel

```
// DrJava stable-20030822
// edu.rice.cs.drjava.ui
// JavadocFrame.java, line 97
private static File _parsePackagesFile(
    File packages, File destDir) {
    try {
        FileReader fr = new FileReader(packages);
        BufferedReader br = new BufferedReader(fr);
        ...
    }
    // fr/br are never closed
}
```

# FindBugs

## Bug Pattern

### Rückgaben von Read prüfen

- `java.io.InputStream` besitzt zwei `read`-Methoden
- Rückgabewert: tatsächlich gelesene Bytes
- Methoden liefern u.U. nicht die erwartete Byteanzahl
- Fehlende Prüfung des Wertes, könnte nicht initialisierte Elemente in den Puffer lesen
- Detektion: linearer Code-Scan

# Beispiel

```
// GNU Classpath 0.06
// java.util
// SimpleTimeZone.java, line 780
int length = input.readInt();
byte[] byteArray = new byte[length];
input.read(byteArray, 0, length);
if (length >= 4){
    ...
}
```

# FindBugs

## Bug Pattern

### Prüfen von Rückgabewerten

- Strings sind in Java keine veränderbaren Objekte
  - ↳ Rückgabe Strings sind immer neue Objekte

Detektion: wie bei read-Methoden

# FindBugs

## Bug Pattern

- Zugriff auf nicht initialisierte Felder in Konstruktoren
- inkonsistente Synchronisation
- Unconditional wait
- wait in Schleifen

# Performance

Analyse von folgenden Applikationen/Bibliotheken:

- GNU Classpath, version 0.06
- rt.jar Sun JDK 1.5.0, build 18
- Eclipse, version 2.1.0
- DrJava, version stable-20030822
- JBoss, version 3.2.2RC3
- jEdit, version 4.1

# Performance

- 1,8 P4 Xeon; 1 GB Speicher
  - max. 13 Minuten um alle Detektoren auf jeder Applikation laufen zu lassen (laut Autor)
- Eigener Test:
  - 1 GH P3; 256 MB Speicher
  - rt.jar aus dem SDK1.4.1\_02
  - mehr als 2h

# Ergebnisse verschiedener Analysen

- Eclipse 2.1.0 (Null Pointer)
- Diverse Bugs in Sun's JDK 1.4.2 und JDK 1.5
- In einer Applikation eines geographischen Informationssystems (200 Pakete, 2000 Klassen und 400,000 Zeilen) wurden 3000 Warnungen produziert. Nach einer Woche wurde die Zahl von den Entwicklern auf 1000 reduziert.

# Beispiel und Zusammenfassung

# Quellen

- <http://findbugs.sourceforge.net>
- <http://pmd.sourceforge.net>