31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Saubere Sache

Clean-Code und Refactoring am praktischen Beispiel

Carsten Siedentop

C/S Consulting

Wer bin ich?

- Carsten Siedentop

- Consultant und Trainer für Java, XML, COBOL, ...
- > 10 Jahre Projekterfahrung in Java
- 17 Jahre Schulungserfahrung
  mit Schwerpunkt auf automatisiertem Testen

- Softwareprodukte
  - PDFUnit
  - PDFMonitor

- COBOL-Buch (in Arbeit)

Agenda

- Was bedeutet eigentlich 'clean'?
- Wer definiert das? Wo?
- Metriken für den Projektalltag? Welche?
- Wie scharf sollten Limits für Metriken sein?
- Wie lebt man 'Clean Code' im Projektalltag?
- Wann ist Refaktoring fertig?

Viele Beispiel life

Refactoring-Ziele

Was heißt eigentlich 'clean' (I)

- Wer definiert das:
  - Projektleiter?
  - Architektur-Team?
  - alle im Projektteam?

Egal?

- Wie ist es definiert:
  - Word-Dokument?
  - Internes Wiki?
  - XML-Datei innerhalb des Projektes?

Egal?

- Gibt es mehrere 'cleans'?

Was heißt eigentlich 'clean' (II)

- Wer definiert das:
  - Projektleiter?
  - Architektur-Team?
  - alle im Projektteam?

Egal?

- Wie ist es definiert:
  - Word-Dokument?
  - Internes Wiki?
  - XML-Datei innerhalb des Projektes?

Egal?

- Gibt es mehrere 'cleans'?

Wie ist das in ihrem Projekt?

Übergeordnete Ziele für guten Code (I)

- KISS keep it simple and stupid
- DRY don't repeat yourself
- SoC Separation of Concerns
- DfCh Design for Change
- DbC Design by Contract
- Test first
- **Red Green Refactor**
- Nur getesteter Code wird eingecheckt, aber mindestens 1x täglich
- YAGNI, You Ain't Gonna Need It

Übergeordnete Ziele für guten Code (II)

- Programmieren Sie klar, seien Sie nicht zu clever
- Sagen Sie das, was Sie wollen, einfach und direkt
- Hören Sie nicht bei Ihrem ersten Entwurf auf
- Benutzen Sie nur die guten Eigenschaften einer Programmiersprache
- Benutzen Sie Bibliotheksfunktionen, wenn möglich
- Verbessern Sie kein schlechtes Programm,
  schreiben Sie es neu.

Übergeordnete Ziele für guten Code (III)

- Code soll einfach zu verstehen sein
- Code soll schnell zu verstehen sein
- Code soll klar sein

Diese Definitionen sind gut, aber unklar

- Public-Methoden müssen mit Javadoc kommentiert sein?

Konkrete Metriken für den Projektalltag

Metriken für den Projektalltag

- Anzahl Zeilen pro Methode < N
- Zeilenlänge < N
- Cyclomatische Komplexität < N
- Anzahl von Übergabeparametern
- Anzahl von Compiler-Warnungen < N
  - z.B. @Override muss gesetzt werden
  - in Eclipse, Maven, Jenkins, etc.
- Keine ternären Operatoren
- if(), for() und while() immer mit Klammern schreiben
- Keine Tabs im Code, nur Spaces

Wie groß ist N?

Keine Metrik für Javadoc

Metriken für den Projektalltag - mit Schwellwert

- Anzahl Zeilen pro Methode < 16
- Zeilenlänge < 120
- Cyclomatische Komplexität < 6
- Anzahl von Übergabeparametern
- Anzahl von Compiler-Warnungen = 0
  - z.B. @Override muss gesetzt werden
  - in Eclipse, Maven, Jenkins, etc.
- Keine ternären Operatoren
- if(), for() und while() immer mit Klammern schreiben
- Keine Tabs im Code, nur Spaces

Keine Metrik für Javadoc

Refactoring-Maßnahmen

- Anzahl Zeilen pro Methode — Code auslagern
- Zeilenlänge — auf mehre Zeilen verteilen
- Cyclomatische Komplexität — Methode verkleinern
- Anzahl von Übergabeparametern — Methode verkleinern
- Anzahl von Compiler-Warnungen — 1. Code fixen, 2. Eclipse etc. konfigurieren
  - z.B. @Override muss gesetzt werden
  - in Eclipse, Maven, Jenkins, etc.
- Keine ternären Operatoren — auf mehrere Zeilen verteilen
- if(), for() und while() immer mit Klammern schreib — Code umschreiben
- Keine Tabs im Code, nur Spaces — Eclipse konfigurieren

Es folgen viele Beispiele

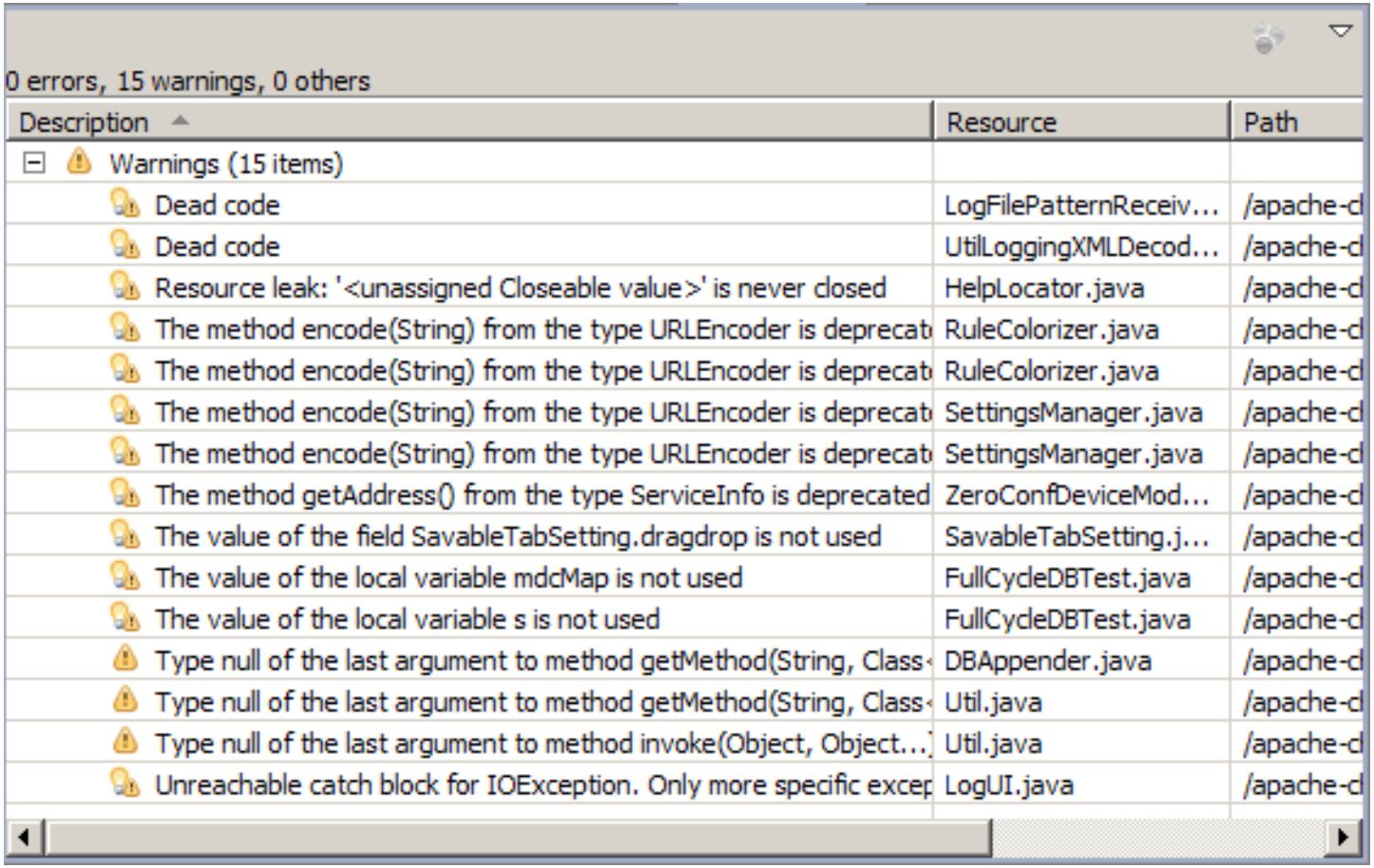
Compiler-Warnungen - vorher

0 errors, 1,488 warnings, 0 others (Filter matched 100 of 1488 items)

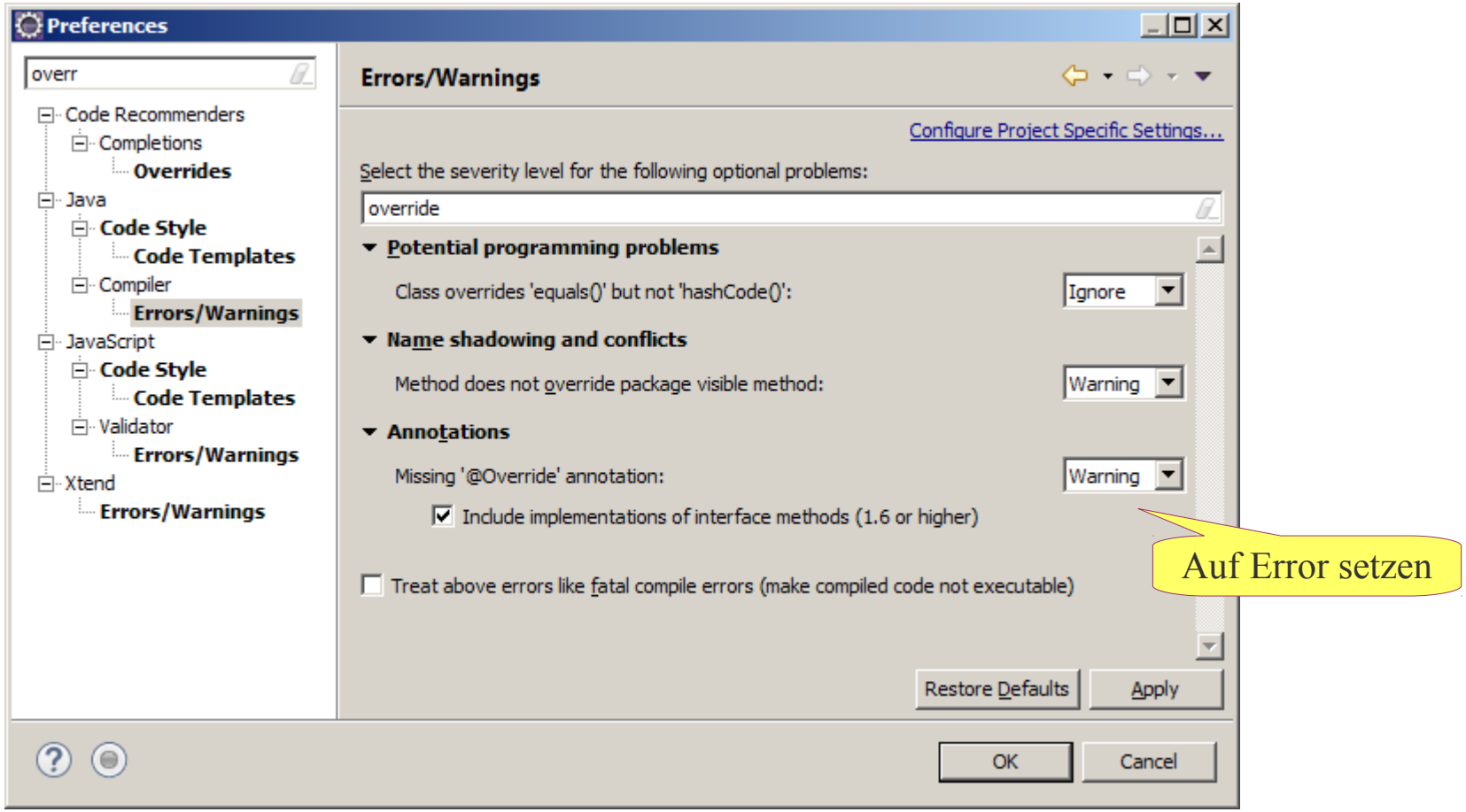
Description	Resource	Path	L
Warnings (100 of 1488 items)			
AbstractList is a raw type. References to generic ty	CyclicBuffe...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	Application...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawA...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawA...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawA...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawC...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawC...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawC...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawC...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawC...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawC...	/apache-chainsa...	li
ArrayList is a raw type. References to generic type	ChainsawC...	/apache-chainsa...	li

Eclipse konfigurieren

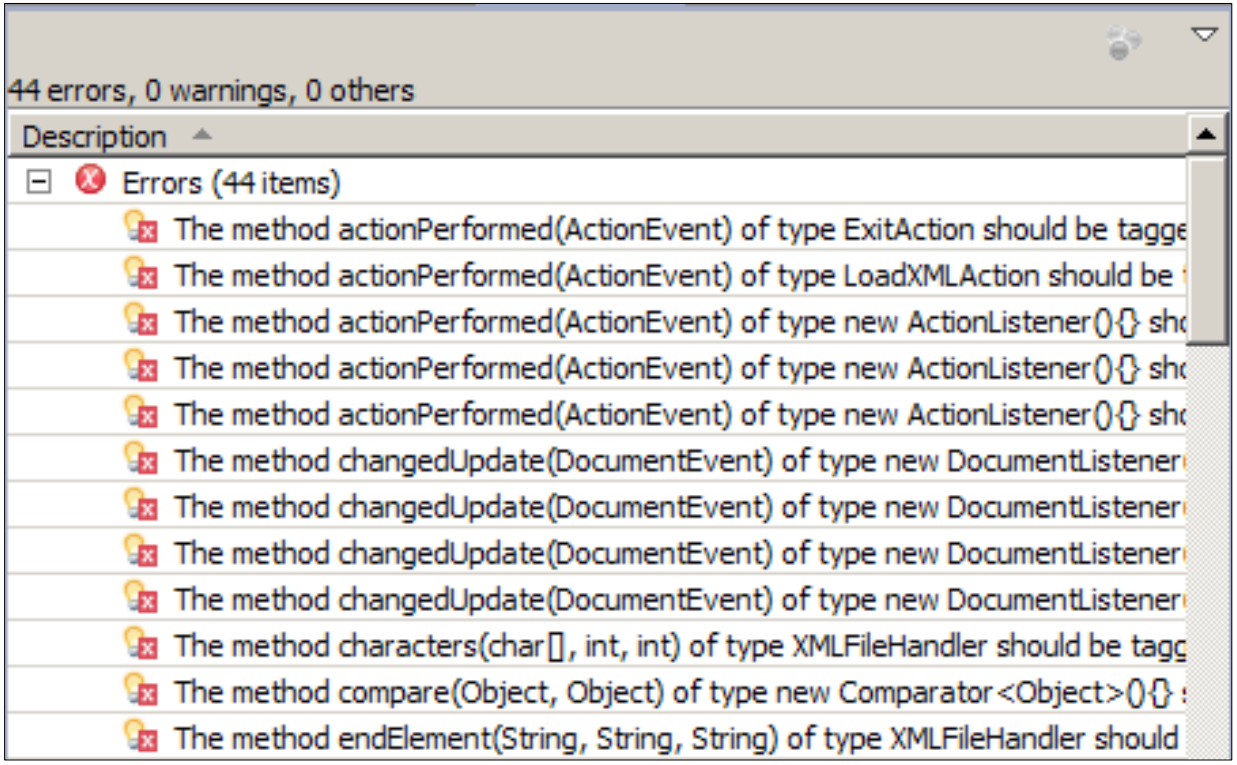
Compiler-Warnungen - nachher



Compiler-Warnungen in Eclipse - vorher



Compiler-Warnungen in Eclipse - nachher



@Override fehlt

Der Compiler ist Ihr Helfer - vorher

```
6  public class DateFormatValidationWithRegexDemo {
7
8    String d1 = "12-03-1973";
9    String d2 = "12.3.1973";
10   String d3 = "12/3/1973";
11   String d4 = "12/03/1973";
12   String d5 = "12. September 1973";
13
14   private String date1 = "20.01.2005";
15   private String date2 = "09.01.2005";
16   private String date3 = "9.1.2005";
17   private String date4 = "31.12.05";
18
```

Kein Problem, oder?

Der Compiler ist Ihr Helfer - nachher

```
 6  public class DateFormatValidationWithRegexDemo {
 7
 8    private String d1 = "12-03-1973";
 9    private String d2 = "12.3.1973";
10    private String d3 = "12/3/1973";
11    private String d4 = "12/03/1973";
12    private String d5 = "12. September 1973";
13
14    private String date1 = "20.01.2005";
15    private String date2 = "09.01.2005";
16    private String date3 = "9.1.2005";
17    private String date4 = "31.12.05";
18
```

unused variable

Kommentare (I) - vorher

```
/**
 * the time of the event
 */
private long mTimeStamp;

/**
 * the priority of the event
 */
private Priority mPriority;

/**
 * the category of the event
 */
private String mCategoryName;
```

Kommentare (I) - nachher

```
private long eventTimeStamp;

private Priority eventPriority;

private String eventCategoryName;
```

Überflüssiger Kommentar ist zweifach vergeudete Zeit

Kommentar (II) - vorher

```
/**
 * @see DefaultHandler
 **/
public void startDocument() throws SAXException {
  numberOfEvents = 0;
}
```

Method Detail

startDocument

```
public void startDocument()
                   throws org.xml.sax.SAXException
```

Specified by:

startDocument in interface org.xml.sax.ContentHandler

Overrides:

startDocument in class org.xml.sax.helpers.DefaultHandler

Throws:

org.xml.sax.SAXException

See Also:

DefaultHandler

Redundanz

Kommentar (II) - nachher

```java
public void startDocument() throws SAXException {
  numberOfEvents = 0;
}
```

Method Detail

startDocument

```
public void startDocument()
                   throws org.xml.sax.SAXException
```

Specified by:

`startDocument in interface org.xml.sax.ContentHandler`

Overrides:

`startDocument in class org.xml.sax.helpers.DefaultHandler`

Throws:

`org.xml.sax.SAXException`

Kommentar (III) - vorher

```
/**
 *
 * Ovverride method getTableCellRendererComponent in DefaultTableCellRenderer.
 *
 * @param aTable
 *          - table object
 * @param aValue
 *          - field value - must be an instace of Priority class
 * @param aIsSelected
 *          - is field selected
 * @param aHasFocus
 *          - is field has focus
 * @param aRow
 *          - field's row number
 * @param aColumn
 *          - field's column number
 * @return instance of PriorityFieldCellRenderer with colors set due to
 *         priority of value
 */
public Component getTableCellRendererComponent(JTable aTable, Object aValue, boolean
  Priority priority = (Priority) aValue;
  Color priorityColor = null;
  if (priority.equals(Priority.FATAL)) {
    priorityColor = Color.red;
```

Wo ist das Wichtige?

Kommentar (III) - nachher

```
/**
 * This method colors a table cell due to the priority of its value.
 */
@Override
public Component getTableCellRendererComponent(JTable aTable, Object aValue,
  Priority priority = (Priority) aValue;
  Color priorityColor = null;
  if (priority.equals(Priority.FATAL)) {
    priorityColor = Color.red;
```

Wann ist ein Kommentar gut?

- Kommentar muss einen Zusatznutzen zum Code haben
- Wiederholen Sie nicht den Code im Kommentar
- Stellen Sie sicher, dass der Kommentar zum Code passt
- Kommentieren Sie kein schlechtes Programm,
  schreiben Sie es neu
- Schreiben Sie Ihren Code so,
  dass er nicht kommentiert werden muss

Gute Namen

Was sind gute Namen?

- Keine semantische Distanz zwischen Namen und Inhalt
- Ein Name darf keinen Interpretationsspielraum haben
- Methodennamen sollen das Ergebnis bennen,
  nicht den Weg dahin.
- Variablennamen benennen den Inhalt
- Konstanten bezeichnen den Wert
- Ähnliche Dinge bekommen ähnliche Namen
- Keine Abkürzungen, außer international übliche
- Namen mit kurzer Gültigkeit können kürzer sein

Methodennamen sind Dokumentation - vorher

```
public static void main(String[] args) {
  TableModel model = new DefaultTableModel(100, 3) {
    @Override
    public Object getValueAt(int row, int column) {
      return "" + (int) Math.pow(row, column + 1);
    }
  };
  JTable table = new JTable(model);
  TableRowSorter<TableModel> rowSorter
              = new TableRowSorter<TableModel>(model);
  table.setRowSorter(rowSorter);


  rowSorter.setComparator(0, new Comparator<String>() {
    public int compare(String s1, String s2) {
      int i1 = Integer.parseInt(s1);
      int i2 = Integer.parseInt(s2);
      return Integer.bitCount(i1) - Integer.bitCount(i2);
    }
  }); ...
```

Nicht klar genug

Methodennamen sind Dokumentation - nachher

```
public static void main(String[] args) {
  TableModel model = createData();




  JTable table = new JTable(model);
  TableRowSorter<TableModel> rowSorter
              = new TableRowSorter<TableModel>(model);
  table.setRowSorter(rowSorter);

  Comparator<String> comparator = createComparator();
  rowSorter.setComparator(0, comparator);




      ...
```

Namen statt Strings - vorher

```
private String createXml(String fileName) {
  StringBuffer buf = new StringBuffer();
  buf.append("<?xml version=\"1.0\" standalone=\"yes\"?>\n");
  buf.append("<!DOCTYPE log4j:eventSet ");
  buf.append("[<!ENTITY data SYSTEM \"file:///");
  buf.append(fileName);
  buf.append("\">]>\n");
  buf.append("<log4j:eventSet xmlns:log4j=\"Claira\">\n");
  buf.append("&data;\n");
  buf.append("</log4j:eventSet>\n");
  return buf.toString();
}
```

Namen statt Strings - nachher

```
private String createXml(String fileName) {
  StringBuffer buf = new StringBuffer();
  buf.append(XML_HEADER);
  buf.append(XML_DOCTYPE_START);
  buf.append(XML_ENTITY_DATA_DECLARATION);
  buf.append(fileName);
  buf.append(XML_DOCTYPE_END);
  buf.append(XML_ROOT_START);
  buf.append(XML_ENTITY_DATA_REFERENCE);
  buf.append(XML_ROOT_END);
  return buf.toString();
}
```

Schneller zu verstehen

Aussagekräftige Namen wählen - vorher

```
public class SchedularDemo_1 {

  public static void main(String[] args) {



    ScheduledExecutorService scheduler =
            Executors.newScheduledThreadPool(1);
    scheduler.scheduleAtFixedRate(new Runnable() {
      public void run() {
        System.out.print(".");
      }
    }, 1, 2, TimeUnit.SECONDS);
  }
}
```

Aussagekräftige Namen wählen - nachher

```
public class SchedularDemo_2 {

  public static void main(String[] args) {
    long initialDelay = 3;
    long interval = 2;
    int poolSize = 1;

    ScheduledExecutorService scheduler =
              Executors.newScheduledThreadPool(poolSize);
    scheduler.scheduleAtFixedRate(new Runnable() {
      public void run() {
        System.out.print(".");
      }
    }, initialDelay, interval, TimeUnit.SECONDS);
  }
}
```

Und nochmal 'magic' - vorher

```
public class UnklareUebergabeparameterDemo_1 {



  @SuppressWarnings("unused")
  private static void demo1() throws IOException {
    File file1 = new File("c:/temp/foo.txt");
    File file2 = new File("c:/temp/bar.txt");
    FileWriter writer1 = new FileWriter(file1, true);
    FileWriter writer2 = new FileWriter(file2, false);
  }

}
```

Was ist append,
was ist overwrite

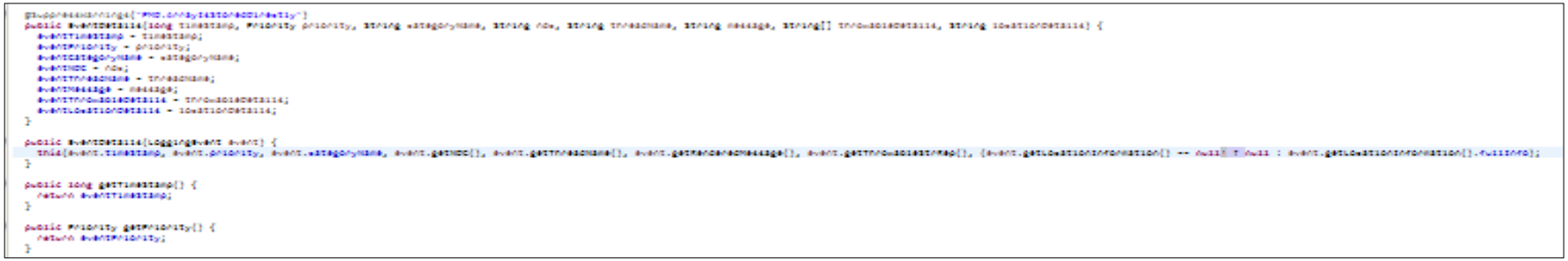
Und nochmal 'magic' - nachher

```
public class UnklareUebergabeparameterDemo_1 {

  private static final boolean FILE_APPEND = true;
  private static final boolean FILE_OVERWRITE = false;

  @SuppressWarnings("unused")
  private static void demo2() throws IOException {
    File file1 = new File("c:/temp/foo.txt");
    File file2 = new File("c:/temp/bar.txt");
    FileWriter fw1 = new FileWriter(file1, FILE_APPEND);
    FileWriter fw2 = new FileWriter(file2, FILE_OVERWRITE);
  }

}
```

Schreibstil

- Formatieren Sie das Programm so,
  dass der Leser im Verständnis unterstützt wird
- Benutzen Sie aussagefähige Variablen- und Funktionsnamen
- Benutzen Sie Einrückungen,
  um die logische Struktur eines Programms zu zeigen
- Setzen Sie Klammern, um Mehrdeutigkeiten zu vermeiden
- Benutzen Sie Leerzeilen, um Befehlsgruppen hervorzuheben
- Machen Sie Ihr Programm von oben nach unten lesbar

Lesbarkeit, lange Zeilen



Sie können das nicht erkennen?

Schlechte Einrückung (I)

```
protected void doMain(String args[], Uphill app, String name) {


        File outputFile = null;
        //ParameterSet params = new ParameterSet();
        //Properties outputProperties = new Properties();
        String outputFileName = null;
    String acronymsFileName=null;
    String inputFileName=null;
    String stateFileName=null;
                // Check the command-line arguments.

        try {
            int i = 0;
            while (true) {
                if (i>=args.length) break;
        if (debug)
            System.err.println("i: "+ i + args.length+" args[i]="+a
```

Schlechte Einrückung (II)

```
                }
        else if (args[i].equals("-i")) {
                    i++;
                    if (args.length < i+1) badUsage(name, "No i
                    inputFileName = args[i++];
        if (debug )
            System.err.println("ip file: "+inputFileName);

                }
        else if (args[i].equals("-s")) {
                    i++;
                    if (args.length < i+1) badUsage(name, "No s
                    stateFileName = args[i++];
        if (debug )
            System.err.println("state file: "+stateFileName);

                }
```

Wenn die Formatierung immer gleich ist, steigt die Lesegeschwindigkeit.

Klammern - ein ewiges Thema (I)

```
/**
 * Attempt to an InputStream. N.B. If an exception is raised, it is ignored.
 *
 * @param is
 */
public static void streamClose( InputStream is )
{
    try
    {
        if( is != null )
        {
            is.close();
        }
    }
    catch( Exception e )
    {
        // do nowt
    }
}
```

Zwei Formatierungen, ...

Klammern - ein ewiges Thema (II)

```
/**
 * Attempt to an InputStream. N.B. If an exception is raised, it is ignored.
 */
public static void streamClose(InputStream is) {
  try {
    if (is != null) {
      is.close();
    }
  } catch (Exception e) {
    // do nowt
  }
}
```

... so erkennt das Auge mehr 'auf einen Blick'

Klammern - noch ein Beispiel

```
/**
 * Get the index of first occurrence of target in source using the offset and count parameters.
 * fromIndex can be used to start beyond zero on source.
 *
 * @return the index of the byte or -1 if not found
 */
static int indexOf(byte[] source, int sourceOffset, int sourceCount, byte[] target, int targetOffset, int targ
    if (fromIndex >= sourceCount) return (targetCount == 0 ? sourceCount : -1);
    if (fromIndex < 0) fromIndex = 0;
    if (targetCount == 0) return fromIndex;

    byte first  = target[targetOffset];
    int max = sourceOffset + (sourceCount - targetCount);

    for (int i = sourceOffset + fromIndex; i <= max; i++) {
        if (source[i] != first) while (++i <= max && source[i] != first);

        if (i <= max) {
            int j = i + 1;
            int end = j + targetCount - 1;
            for (int k = targetOffset + 1; j < end && source[j] == target[k]; j++, k++);

            if (j == end) return i - sourceOffset;
        }
    }
    return -1;
}
```

Das geht auch kürzer ;-)

?

?

Immer klammern?

sslKeyExchange.c

```
624             goto fail;
625         if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
626             goto fail;
627         if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
628             goto fail;
629         if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
630             goto fail;
631             goto fail;
632         if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
633             goto fail;
634
635         err = sslRawVerify(ctx,
636                            ctx->peerPubKey,
637                            dataToSign,         /* plaintext */
638                            dataToSignLen,      /* plaintext length */
639                            signature,
640                            signatureLen);
641     if(err) {
642         sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
643                     "returned %d\n", (int)err);
644         goto fail;
645     }
```

Immer klammern!

sslKeyExchange_better.c

```
625             goto fail;
626         }
627         if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0) {
628             goto fail;
629         }
630         if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0) {
631             goto fail;
632         }
633         if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0) {
634             goto fail;
635         }
636         goto fail;
637         if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0) {
638             goto fail;
639         }
640
641         err = sslRawVerify(ctx,
642                            ctx->peerPubKey,
643                            dataToSign,         /* plaintext */
644                            dataToSignLen,      /* plaintext length */
645                            signature,
646                            signatureLen);
647     if(err) {
648         sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
649                     "returned %d\n", (int)err);
650         goto fail;
651     }
```

Mit Klammern wäre der Fehler schneller aufgefallen

Sauber einrücken - ohne Tabs (I)

```
1314      */
1315     protected synchronized final void writeLock() {
1316   try {
1317       while ((numReaders > 0) || (currWriter != null)) {
1318     if (Thread.currentThread() == currWriter) {
1319                         if (notifyingListeners) {
1320                             // Assuming one doesn't do something wrong in a
1321                             // subclass this should only happen if a
1322                             // DocumentListener tries to mutate the document.
1323                             throw new IllegalStateException(
1324                                     "Attempt to mutate in notification");
1325                         }
1326                         numWriters++;
1327                         return;
1328                     }
1329     wait();
1330       }
1331       currWriter = Thread.currentThread();
1332             numWriters = 1;
1333   } catch (InterruptedException e) {
1334       throw new Error("Interrupted attempt to aquire write lock");
1335   }
1336     }
1337
```

Sauber einrücken - ohne Tabs (II)

```
1314 ·····*/¶
1315 ····protected·synchronized·final·void·writeLock()·{¶
1316 » try·{¶
1317 » ····while·((numReaders·>·0)·||·(currWriter·!=·null))·{¶
1318 » » if·(Thread.currentThread()·==·currWriter)·{¶
1319 ····················if·(notifyingListeners)·{¶
1320 ························//·Assuming·one·doesn't·do·something·wrong·in·a¶
1321 ························//·subclass·this·should·only·happen·if·a¶
1322 ························//·DocumentListener·tries·to·mutate·the·document.¶
1323 ························throw·new·IllegalStateException(¶
1324 ····························"Attempt·to·mutate·in·notification");¶
1325 ····················}¶
1326 ····················numWriters++;¶
1327 ····················return;¶
1328 ················}¶
1329 » » wait();¶
1330 » ····}¶
1331 » ····currWriter·=·Thread.currentThread();¶
1332 ············numWriters·=·1;¶
1333 » }·catch·(InterruptedException·e)·{¶
1334 » ····throw·new·Error("Interrupted·attempt·to·aquire·write·lock");¶
1335 » }¶
1336 ····}¶
1337 ¶
```

Unnötiges 'final' - vorher

```
final ListSelectionModel lsm = (ListSelectionModel) aEvent.getSource();
if (lsm.isSelectionEmpty()) {
  detailsPane.setText("Nothing selected");
} else {
  final int selectedRow = lsm.getMinSelectionIndex();
  final EventDetails e = model.getEventDetails(selectedRow);
  final Object[] args = { new Date(e.getTimeStamp()), e.getPriority(), e
      escape(e.getCategoryName()), escape(e.getLocationDetails()), escap
  detailsPane.setText(FORMATTER.format(args));
  detailsPane.setCaretPosition(0);
}
```

Unnötiges 'final' - was das Auge sucht

```
      ListSelectionModel lsm = (ListSelectionModel) aEvent.getSource();
if (lsm.isSelectionEmpty()) {
  detailsPane.setText("Nothing selected");
} else {
        int selectedRow = lsm.getMinSelectionIndex();
        EventDetails e = model.getEventDetails(selectedRow);
        Object[] args = { new Date(e.getTimeStamp()), e.getPriority(), e
      escape(e.getCategoryName()), escape(e.getLocationDetails()), escap
  detailsPane.setText(FORMATTER.format(args));
  detailsPane.setCaretPosition(0);
}
```

Unnötiges 'final' - so ist es besser zu lesen

```
ListSelectionModel lsm = (ListSelectionModel) aEvent.getSource();
if (lsm.isSelectionEmpty()) {
  detailsPane.setText("Nothing selected");
} else {
  int selectedRow = lsm.getMinSelectionIndex();
  EventDetails e = model.getEventDetails(selectedRow);
  Object[] args = { new Date(e.getTimeStamp()), e.getPriority(),
      escape(e.getCategoryName()), escape(e.getLocationDetails())
  detailsPane.setText(FORMATTER.format(args));
  detailsPane.setCaretPosition(0);
}
```

Lesbarkeit, ternäre Operatoren

Ternärer Operator - vorher

```
super.setForeground((unselectedForeground != null) ? unselectedForeground : table.getForeground());
```

Ternärer Operator - nachher

```
Color foregroundColor = table.getForeground();
if (unselectedForeground != null) {
  foregroundColor = unselectedBackground;
}
super.setForeground(foregroundColor);
```

Ternärer Operator, Prüfung auf 'null' - vorher

```
public ForeignKeyMeta(Node foreignKeyNode) {
  NamedNodeMap attribs = foreignKeyNode.getAttributes();

  Node node = attribs.getNamedItem("table");
  if (node == null)
    throw new IllegalStateException("'table' attribute required");
  tableName = node.getNodeValue();

  node = attribs.getNamedItem("column");
  if (node == null)
    throw new IllegalStateException("'column' attribute required");
  columnName = node.getNodeValue();

  node = attribs.getNamedItem("remoteSchema");
  remoteSchema = node == null ? null : node.getNodeValue();

  node = attribs.getNamedItem("remoteCatalog");
  remoteCatalog = node == null ? null : node.getNodeValue();
}
```

Wie schnell ist dieser Code zu verstehen?

Ternärer Operator, Prüfung auf 'null' - nachher

```
public ForeignKeyMeta(Node foreignKeyNode) {
  assertAttributeAvaible(foreignKeyNode, "table" );
  assertAttributeAvaible(foreignKeyNode, "column");

  tableName      = getAttributeValue(foreignKeyNode, "table");
  columnName     = getAttributeValue(foreignKeyNode, "column");
  remoteSchema   = getAttributeValue(foreignKeyNode, "remoteSchema");
  remoteCatalog  = getAttributeValue(foreignKeyNode, "remoteCatalog");

  logCurrentForeignKeyMetadata();
}
```

Ternärer Operator, weitere Beispiele (I)

```
container = schema != null ? schema : catalog != null ? catalog : db.getName();

String parentContainer = pkSchema != null ? pkSchema : pkCatalog != null ?
                         pkCatalog : db.getName();

String baseContainer = config.getSchema() != null ? config.getSchema() :
                       config.getCatalog() != null ? config.getCatalog() :
                       db.getName();

String fullName = (catalog == null && schema == null ? db + '.' : "") +
                  (catalog == null ? "" : catalog + '.') +
                  (schema == null ? "" : schema + '.') + table;

public void setComments(String comments) {
  String cmts = (comments == null || comments.trim().length() == 0)
                ? null : comments.trim();
  if (cmts != null) {
    cmts = cmts.substring(0, crapIndex).trim();
    cmts = cmts.length() == 0 ? null : cmts;
  }
  this.comments = cmts;
}
```

Wehret den Anfängen, ...

Ternärer Operator, weitere Beispiele (II)

DirectByteBuffer.class

```
620
621     public ShortBuffer asShortBuffer() {
622         int off = this.position();
623         int lim = this.limit();
624         assert (off <= lim);
625         int rem = (off <= lim ? lim - off : 0);
626
627         int size = rem >> 1;
628         if (!unaligned && ((address + off) % (1 << 1) != 0)) {
629             return (bigEndian
630                     ? (ShortBuffer)(new ByteBufferAsShortBufferB(this,
631                                                                  -1,
632                                                                  0,
633                                                                  size,
634                                                                  size,
635                                                                  off))
636                     : (ShortBuffer)(new ByteBufferAsShortBufferL(this,
637                                                                  -1,
638                                                                  0,
639                                                                  size,
640                                                                  size,
641                                                                  off)));
642         } else {
643             return (nativeByteOrder
644                     ? (ShortBuffer)(new DirectShortBufferU(this,
645                                                            -1,
646                                                            0,
647                                                            size,
648                                                            size,
649                                                            off))
650                     : (ShortBuffer)(new DirectShortBufferS(this,
651                                                            -1,
652                                                            0,
653                                                            size,
654                                                            size,
655                                                            off)));
656         }
657     }
```

... sonst endet es so

Lesbarkeit, anonyme Klassen

Anonyme Klassen extrahieren (I) - vorher

```
JButton toggleButton = new JButton("Pause");
toggleButton.setMnemonic('p');
toggleButton.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent aEvent) {
    aModel.toggle();
    if (aModel.isPaused()) {
      toggleButton.setText("Resume");
    } else {
      toggleButton.setText("Pause");
    }
  }
});
gridbag.setConstraints(toggleButton, c);
add(toggleButton);
```

Lokale Variable erstellen

Anonyme Klassen extrahieren (I) - zwischendrin

```
JButton toggleButton = new JButton("Pause");
toggleButton.setMnemonic('p');
ActionListener togglePauseAction = new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent aEvent) {
    aModel.toggle();
    if (aModel.isPaused()) {
      toggleButton.setText("Resume");
    } else {
      toggleButton.setText("Pause");
    }
  }
};
toggleButton.addActionListener(togglePauseAction);
gridbag.setConstraints(toggleButton, c);
add(toggleButton);
```

Externe Klasse erstellen

Anonyme Klassen extrahieren (I) - nachher

```
JButton toggleButton = new JButton("Pause");
toggleButton.setMnemonic('p');
ActionListener togglePauseAction = new TogglePauseAction();
toggleButton.addActionListener(togglePauseAction);
gridbag.setConstraints(toggleButton, c);
add(toggleButton);
```

Anonyme Klasse extrahieren (II) - vorher

```
constraints.gridy++;
JTextField msgField = MessageTextField.getInstance();
DocumentListener messageDocumentListener = new DocumentListener() {
  @Override
  public void insertUpdate(DocumentEvent aEvent) {
    aModel.setMessageFilter(msgField.getText());
  }

  @Override
  public void removeUpdate(DocumentEvent aEvent) {
    aModel.setMessageFilter(msgField.getText());
  }

  @Override
  public void changedUpdate(DocumentEvent aEvent) {
    aModel.setMessageFilter(msgField.getText());
  }
};
msgField.getDocument().addDocumentListener(messageDocumentListener);
gridbag.setConstraints(msgField, constraints);
add(msgField);
```

Bezieht sich nur auf msgField

Anonyme Klasse extrahieren (II) - nachher

```
constraints.gridy++;
JTextField msgField = MessageTextField.getInstance();
gridbag.setConstraints(msgField, constraints);
add(msgField);
```

Assign null - (1)

```
@Override
public void endElement(String aNamespaceURI, String aLocalName, String aQName) {
  if (TAG_EVENT.equals(aQName)) {
    addEvent();
    resetData();
  } else if (currentElement != TAG_EVENT) {
    currentElement = TAG_EVENT; // hack - but only thing I care about
  }
}

private void addEvent() {
  mModel.addEvent(new EventDetails(eventTimeStamp, eventPriority, eventCategoryName, eve
  numberOfEvents++;
}

private void resetData() {
  eventTimeStamp = 0;
  eventPriority = null;
  eventCategoryName = null;
  eventNDC = null;
  eventThreadName = null;
  eventMessage = null;
  eventThrowableDetails = null;
  eventLocationDetails = null;
}
```

Assign null - (2)

```
@Override
public void endElement(String aNamespaceURI, String aLocalName, String aQName) {
  if (TAG_EVENT.equals(aQName)) {
    addEvent();
    resetData();
  } else if (currentElement != TAG_EVENT) {
    currentElement = TAG_EVENT; // hack - but only thing I care about
  }
}

private void addEvent() {
  currentLogEvent.setEventTimeStamp(eventTimeStamp);
  currentLogEvent.setEventPriority(eventPriority);
  currentLogEvent.setEventCategoryName(eventCategoryName);
  currentLogEvent.setEventNDC(eventNDC);
  currentLogEvent.setEventThreadName(eventThreadName);
  currentLogEvent.setEventMessage(eventMessage);
  currentLogEvent.setEventThrowableDetails(eventThrowableDetails);
  currentLogEvent.setEventLocationDetails(eventLocationDetails);
  mModel.addEvent(currentLogEvent);
  numberOfEvents++;
}

private void resetData() {
  this.currentLogEvent = new EventDetails();
}
```

Assign null - (3)

```
@Override
public void endElement(String aNamespaceURI, String aLocalName, String aQName) {
  if (TAG_EVENT.equals(aQName)) {
    addEvent();
  } else if (currentElement != TAG_EVENT) {
    currentElement = TAG_EVENT; // hack - but only thing I care about
  }
}

private void addEvent() {
  currentLogEvent.setEventTimeStamp(eventTimeStamp);
  currentLogEvent.setEventPriority(eventPriority);
  currentLogEvent.setEventCategoryName(eventCategoryName);
  currentLogEvent.setEventNDC(eventNDC);
  currentLogEvent.setEventThreadName(eventThreadName);
  currentLogEvent.setEventMessage(eventMessage);
  currentLogEvent.setEventThrowableDetails(eventThrowableDetails);
  currentLogEvent.setEventLocationDetails(eventLocationDetails);
  mModel.addEvent(currentLogEvent);
  numberOfEvents++;
}
```

moved into startElement()

Assign null - fertig, startElement()

```
@SuppressWarnings("PMD.ExcessiveParameterList")
@Override
public void startElement(String namespaceURI, String localName, String name, Attributes att
  switch (name) {
  case TAG_EVENT:
    currentLogEvent = new EventDetails();
    String eventThreadName = attrs.getValue("thread");
    long eventTimeStamp = Long.parseLong(attrs.getValue("timestamp"));
    String eventCategoryName = attrs.getValue("category");
    Priority eventPriority = Priority.toPriority(attrs.getValue("priority"));
    currentLogEvent.setEventThreadName(eventThreadName);
    currentLogEvent.setEventTimeStamp(eventTimeStamp);
    currentLogEvent.setEventCategoryName(eventCategoryName);
    currentLogEvent.setEventPriority(eventPriority);
    break;
  case TAG_LOCATION_INFO:
    String eventLocationDetails = attrs.getValue("class") + "." + attrs.getValue("method")
    currentLogEvent.setEventLocationDetails(eventLocationDetails);
    break;
  case TAG_NDC:
```

Assign null - fertig, endElement()

```
@Override
public void endElement(String namespaceURI, String localName, String qName) {
  if (TAG_EVENT.equals(qName)) {
    model.addEvent(currentLogEvent);
    numberOfEvents++;
  } else {
    currentElement = TAG_OF_NO_INTEREST;
  }
}
```

Geschachtelte if's - (I)

```
@Override
public Object getValueAt(int row, int col) {
  synchronized (synchronizationObject) {
    EventDetails event = filteredEvents[row];

    if (col == 0) {
      return DATE_FORMATTER.format(new Date(event.getTimeStamp()));
    } if (col == 1) {
      return event.getPriority();
    } else if (col == 2) {
      if (event.getThrowableStrRep() == null) {
        return Boolean.FALSE;
      } else {
        return Boolean.TRUE;
      }
    } else if (col == 3) {
      return event.getCategoryName();
    } else if (col == 4) {
      return event.getNDC();
    }
    return event.getMessage();
  }
}
```

Geschachtelte if's - (II)

```
@Override
public Object getValueAt(int row, int col) {
  synchronized (synchronizationObject) {
    EventDetails event = filteredEvents[row];

    if (col == 0) {
      return DATE_FORMATTER.format(new Date(event.getTimeStamp()));
    }
    if (col == 1) {
      return event.getPriority();
    }
    if (col == 2) {
      if (event.getThrowableStrRep() == null) {
        return Boolean.FALSE;
      } else {
        return Boolean.TRUE;
      }
    }
    if (col == 3) {
      return event.getCategoryName();
    }
    if (col == 4) {
      return event.getNDC();
    }
    return event.getMessage();
  }
}
```

Geschachtelte if's - (III)

```
@Override
public Object getValueAt(int row, int col) {
  synchronized (synchronizationObject) {
    EventDetails event = filteredEvents[row];
    switch (col) {
    case 0:
      return DATE_FORMATTER.format(new Date(event.getTimeStamp()));
    case 1:
      return event.getPriority();
    case 2:
      if (event.getThrowableStrRep() == null) {
        return Boolean.FALSE;
      } else {
        return Boolean.TRUE;
      }
    case 3:
      return event.getCategoryName();
    case 4:
      return event.getNDC();
    default:
      return event.getMessage();
    }
  }
}
```

Nochmal geschachtelt if's - (I)

```
@Override
public void characters(char[] aChars, int aStart, int aLength) {
  if (mCurrentElement == TAG_NDC) {
    eventNDC = new String(aChars, aStart, aLength);
  } else if (mCurrentElement == TAG_MESSAGE) {
    eventMessage = new String(aChars, aStart, aLength);
  } else if (mCurrentElement == TAG_THROWABLE) {
    final StringTokenizer st = new StringTokenizer(new String(aChars, aStart, aLength), "\t");
    eventThrowableDetails = new String[st.countTokens()];
    if (eventThrowableDetails.length > 0) {
      eventThrowableDetails[0] = st.nextToken();
      for (int i = 1; i < eventThrowableDetails.length; i++) {
        eventThrowableDetails[i] = "\t" + st.nextToken();
      }
    }
  }
}
```

Nochmal geschachtelt if's - (II)

```
@Override
public void characters(char[] aChars, int aStart, int aLength) {
  String currentData = new String(aChars, aStart, aLength);
  switch (mCurrentElement) {
  case TAG_NDC:
    eventNDC = currentData;
    break;
  case TAG_MESSAGE:
    eventMessage = currentData;
  case TAG_THROWABLE:
    eventThrowableDetails = StringHelper.tokenizeByTab(currentData);
  }
}
```

Komplexe Zeile

```
needUpdate = needUpdate || matchFilter(event);
```

Das geht einfacher

```
if (matchFilter(event)) {
  needUpdate = true;
}
```

```
needUpdate = updateWhenMatchingEvents(event);
```

so!

```
private boolean updateWhenMatchingEvents(EventDetails event) {
  if (matchFilter(event)) {
    return true;
  }
  return false;
}
```

Konfuses Errorhandling - Wo ist das Konzept?

```
public static void main(String[] argv) throws Exception {
  if (argv.length == 1 && argv[0].equals("-gui")) { // warning: serious temp hack
      new MainFrame().setVisible(true);
      return;
  }
  SchemaAnalyzer analyzer = new SchemaAnalyzer();
  int rc = 1;
  try {
      rc = analyzer.analyze(new Config(argv)) == null ? 1 : 0;
  } catch (ConnectionFailure couldntConnect) {
      // failure already logged
      rc = 3;
  } catch (EmptySchemaException noData) {
      // failure already logged
      rc = 2;
  } catch (InvalidConfigurationException badConfig) {
      System.err.println();
      if (badConfig.getParamName() != null)
          System.err.println("Bad parameter specified for " + badConfig.getParamName());
      System.err.println(badConfig.getMessage());
      if (badConfig.getCause() != null && !badConfig.getMessage().endsWith(badConfig.getMessage()))
          System.err.println(" caused by " + badConfig.getCause().getMessage());
      Logger logger = Logger.getLogger(Main.class.getName());
      logger.log(Level.FINE, "Command line parameters: " + Arrays.asList(argv));
      logger.log(Level.FINE, "Invalid configuration detected", badConfig);
  } catch (ProcessExecutionException badLaunch) {
      System.err.println(badLaunch.getMessage());
  } catch (Exception exc) {
      exc.printStackTrace();
  }
  System.exit(rc);
}
```

Konfuses Errorhandling - Knackpunkte

```
public static void main(String[] argv) throws Exception {
  if (argv.length == 1 && argv[0].equals("-gui")) { // warning: serious temp hack
      new MainFrame().setVisible(true);
      return;
  }
  SchemaAnalyzer analyzer = new SchemaAnalyzer();
  int rc = 1;
  try {
      rc = analyzer.analyze(new Config(argv)) == null ? 1 : 0;
  } catch (ConnectionFailure couldntConnect) {
      // failure already logged
      rc = 3;
  } catch (EmptySchemaException noData) {
      // failure already logged
      rc = 2;
  } catch (InvalidConfigurationException badConfig) {
      System.err.println();
      if (badConfig.getParamName() != null)
          System.err.println("Bad parameter specified for
      System.err.println(badConfig.getMessage());
      if (badConfig.getCause() != null && !badConfig.getMessage().endsWith(badConfig.getMessage()))
          System.err.println(" caused by " + badConfig.getCause().getMessage());
      Logger logger = Logger.getLogger(Main.class.getName());
      logger.log(Level.FINE, "Command line parameters: " + Arrays.asList(argv));
      logger.log(Level.FINE, "Invalid configuration detected", badConfig);
  } catch (ProcessExecutionException badLaunch) {
      System.err.println(badLaunch.getMessage());
  } catch (Exception exc) {
      exc.printStackTrace();
  }
  System.exit(rc);
}
```

RC für das Betriebssystem

Hier sollte geloggt werden

Zusätzliche Benutzung der Konsole

Kein RC

Kein Stacktrace, kein RC

Kein Logging, kein RC

Errorhandling - So ist's auch nicht gut

```
277                 retval.set(Calendar.MILLISECOND, 0);
278             }
279             catch( NumberFormatException e )
280             {
281                 for( int i=0; retval == null && i<POTENTIAL_FORMATS.length; i++
282                 {
283                     try
284                     {
285                         Date utilDate = POTENTIAL_FORMATS[i].parse( date );
286                         retval = new GregorianCalendar();
287                         retval.setTime( utilDate );
288                     }
289                     catch( ParseException pe )
290                     {
291                         //ignore and move to next potential format
292                     }
293                 }
294                 if( retval == null )
295                 {
296                     //we didn't find a valid date format so throw an exception
297                     throw new IOException( "Error converting date:" + date );
298                 }
299             }
300         }
301         return retval;
```

Clean-Code im Kontext von Tests

Code testbar gestalten

- Kleine Methoden sind besser testbar
- Keine heimlichen Annahmen (zahl > 0)
- Interfaces ermöglichen den Einsatz von Mock-Klassen
- Halten Sie die Abhängigkeiten von Methoden und Klassen klein
- Zustandsbehaftete Klassen sind schwerer zu testen
- Wenn Sie viele Tests schreiben,
  werden Sie „automatisch“ testbaren Code schreiben

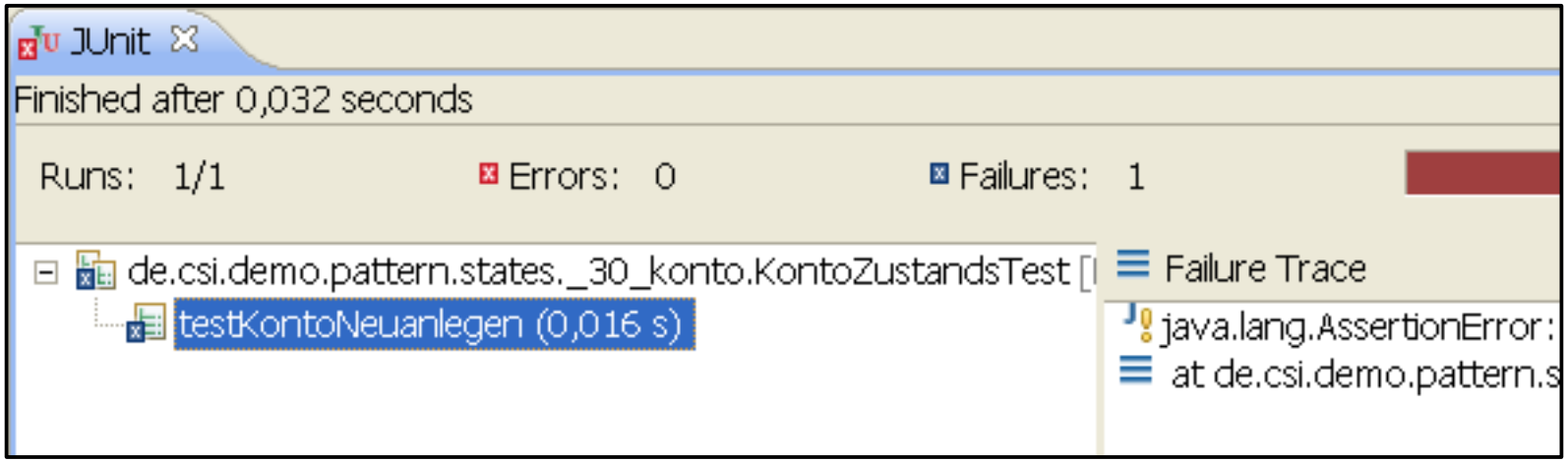
Schlechte und gute Tests - vorher

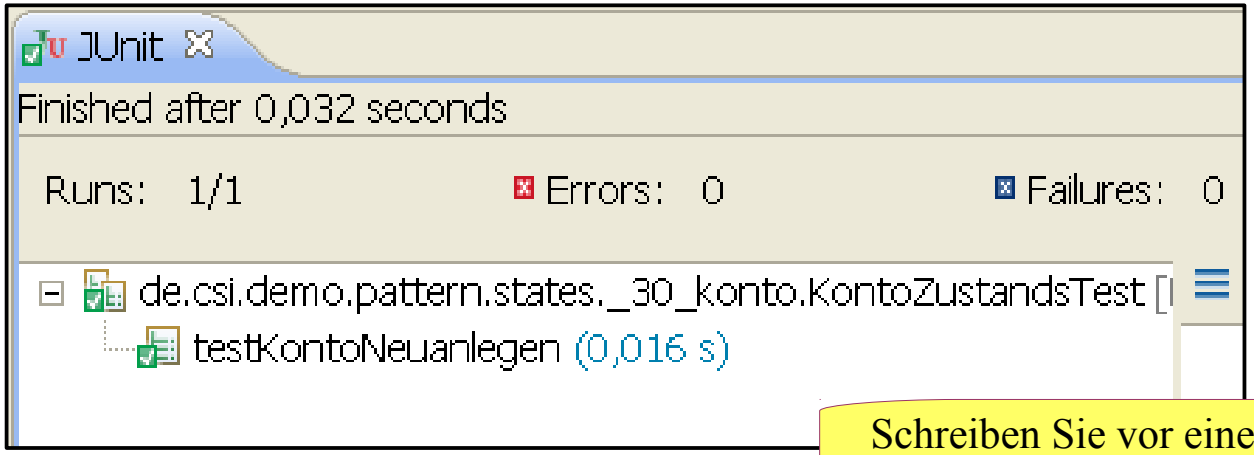
```
@Test
public void testParseFIXMEs() {
  String line = "//FIXME test 1";
  Task parsedLine = fixmeTagParser.parse(line);
  assertEquals(" test 1", parsedLine.getComment());

  line = "//FIXME test 2";
  parsedLine = fixmeTagParser.parse(line);
  assertEquals(" test 2", parsedLine.getComment());

  line = "/*FIXME test 3*/";
  parsedLine = fixmeTagParser.parse(line);
  assertEquals(" test 3*/", parsedLine.getComment());

  line = "* FIXME test 4";
  parsedLine = fixmeTagParser.parse(line);
  assertEquals(" test 4", parsedLine.getComment());

  line = "#FIXME test 5";
  parsedLine = fixmeTagParser.parse(line);
  assertEquals(" test 5", parsedLine.getComment());
}
```

Schlechte und gute Tests - nachher

```java
@Test
public void parseBlockComment_FIXME_PositionAfterSomeOtherComment() {
  String line = "/* abcde FIXME comment_here */";
  String expected = "comment_here";
  String actual = fixmeTagParser.parse(line).getComment();
  assertEquals(expected, actual);
}

@Test
public void parseBlockComment_FIXME_CodeBeforeComment() {
  String line = "code_here /*FIXME comment_here*/";
  String expected = "comment_here";
  String actual = fixmeTagParser.parse(line).getComment();
  assertEquals(expected, actual);
}

@Test
public void parseBlockComment_FIXME_EmptyComment() {
  String line = "/*FIXME*/";
  String expected = "";
  String actual = fixmeTagParser.parse(line).getComment();
  assertEquals(expected, actual);
}
```

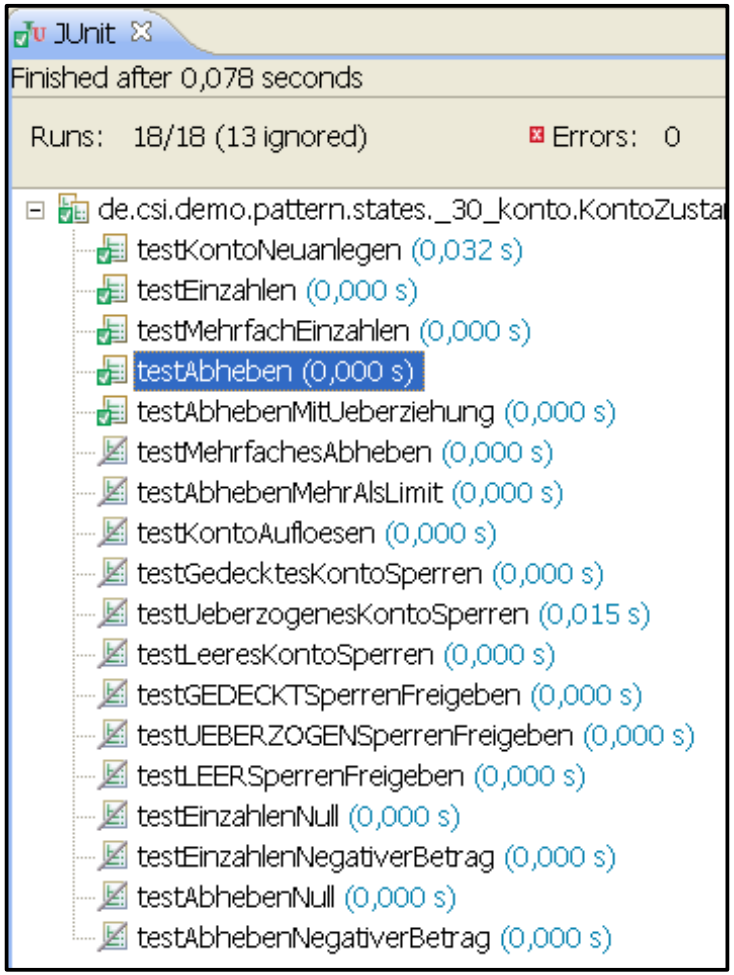
Wann wird refactor'd - Red Green Refactor





Schreiben Sie vor einem Refactoring Tests

Tests mit @IGNORE markieren



Kein try/catch in Tests

Vorher

```
28  public void testTransformation(){
    Source xml = null;
    Source xsl = null;
31  try {
32    File xsltFile = new File("test"+File.separator+"XsltCoverageStats.xsl");
33    xml = new StreamSource("test"+File.separator+"coverage.xml");
34    //use always the URL of the XSLT
35    xsl =  new StreamSource(xsltFile.toURI().toURL().toString());
36  } catch (MalformedURLException e) {
37    // TODO Auto-generated catch bl
38    e.printStackTrace();
39  }
40  try {
41    String result = doXSLT(xml, xsl);
42    assertNotNull(result);
43  } catch (Exception e) {
44    fail("Exception occured: " + e.getMessage());
45  }
46  }
```

schlecht

Stacktrace der Exception wird vernichtet

Nachher

```
29  public void testTransformation() throws Exception {
30    String xmlFilename = "test/coverage.xml";
31    String xslFilename = "test/XsltCoverageStats.xsl";
32    String result = transformXSLT(xmlFilename, xslFilename);
33    assertNotNull(result);
34  }
35
```

Clean-Code und Performance-Tuning

Sauber tunen

- Messen Sie, bevor Sie tunen
- Opfern Sie niemals Klarheit für Effizienz
- Machen Sie es richtig, bevor Sie es schneller machen
  Machen Sie es klar, bevor Sie es schneller machen
  Halten Sie es richtig und klar, wenn Sie es schneller machen
- Überlassen Sie dem Compiler die kleinen Optimierungen
- Bestehen Sie nicht auf bestehendem Code,
  Reorganisieren Sie statt dessen

Das Ende vom Refactoring

Ende der Code-Verbesserung

- Wenn der Compiler keine Warnungen ausgibt?

- Wenn Klassen oder Methoden klein genug sind?
- Wenn eine Klasse nicht zu viele Methoden besitzt?
- Wenn die Namen von Klassen/ Methoden gut sind?

- Wenn Ihr Code 'gut' aussieht?
- Wenn der Code auf Anhieb zu verstehen ist?

Wenn Sie Ihr Ziel erreicht haben!

Ist Clean-Code wirtschaftlich sinnvoll

Ist Clean-Code wirtschaftlich sinnvoll?

Ja

Viel Erfolg mit sauberem Code!

Carsten Siedentop


Fragen, Anregungen, Kritik

Viel Erfolg mit sauberem Code!

Carsten Siedentop