Albrecht Weinert

Zehn Schritte nach JAVA 1.5.0

Von 1.4.2 + Eclipse nach 1.5.0 + NetBeans

Prof. Dr.-Ing. Albrecht Weinert

Labor für Medien und verteilte Anwendungen (*MEVA*-Lab)

Fachbereich 3 Elektrotechnik und Informatik
Fachhochschule Bochum

Dokument: Java15undNB0p-fPDFmaker.sxw (Vorlage), erstellt am 11. November 2004

Version	Datum	Autor	Änderung / Anlass
V 0 . 0	12.11.2004	Weinert	neu
V 0 . 1	21.11.2004	We	Korrekturen
V 0 . 2	23.12.2004	We	kl. Ergnz.; erste Fass f. .pdf. Internet

Letzte Änderung am 23.12.04, 13:12 (letztes Speicherdatum)

Inhaltsverzeichnis

1 Aufbruch nach Java 1.5 .

Der gewaltige „Tiger“-Sprung von <= 1.4.x nach >= 1.5.0 wurde schon an vielen Stellen kommentiert und wird wohl noch einige Zeit „das“ große Thema für die Java-Gemeinde sein. Java 1.5 liefert nun ja nun praktisch alles, was einem bisher bei der Sprache Java noch fehlte.

Das Folgende beschäftigt sich nun gerade nicht mit den zusätzlichen Möglichkeiten, die 1.5 bietet. Es geht um die Installation und Einrichtung von 1.5.0 und dabei besonders um die Frage, ob und wieweit bisher – also unter <=1.4.x – verwendete Frameworks, Klassenbibliotheken, Erweiterungen und Werkzeuge unverändert auch in 1.5 installiert bzw. weiterverwendet werden können.

Nach dem Lobgesang über die neuen 1.5-Möglichkeiten scheint diese Frage auf den ersten Blick vielleicht etwas unpassend. Für Entwicklungen ist sie aber äquivalent mit der Frage, wieweit man Kunden mit einer Plattform <= 1.4.x und (zur Zeit noch seltenere) mit Plattform >= 1.5 mit denselben . jar-Files, installed extensions etc. beliefern kann. Die folgende Untersuchung dazu mag auch die eine oder Anregung zur Handhabung von Java-Installationen und -Erweiterungen enthalten.

1.1 JDK1.5.0 (final) als Anlass .

Seit wenigen Tagen ist nach einigen „early accesses“ und „betas“ nun die „final“ Version des Java SDK (JDK) / JRE 1.5.0 von SUN herausgegeben worden. Das Ganze wird unter Unterdrückung der führenden „1.“ auch Java 5.0 genannt (Zeitpunkt des Schreibens 11.11.2004).

Das vielfach ersehnte Erscheinen des final wird der Start für das Portieren von Projekten von 1.4. nach 1.5 sein, und es war auch der Anlass für die eingangs gestellte Kompatibilitätsfrage mit bisherigen Tools und Erweiterungen.

Um die wesentlichen Ergebnisse vorwegzunehmen: Trotz des Riesenschrittes nach vorne funktioniert fast alles Bewährte klaglos weiter. Die Wermutstropfen sind: Eclipse 3.0 verträgt den Schritt nach 1.5 nicht (jedenfalls so einfach ohne Zusätze mit Kolateralschäden) und JavaDoc hat auch in der final-Version einen alten beim Schritt nach 1.5 erneut aufgetauchten (sogenannten „regression“-) Bug, der (solange keine 1.5-Spracherweiterungen verwendet werden) mit dem Rückgriff auf JavaDoc 1.4 umgangen werden kann.

Für eine lokale Installation des JDK1.5.0 und der Dokumentation benötigt man zwei der in Listing 1 gezeigten Dateien. Alle im Folgenden erwähnten Werkzeuge und Klassenbibliotheken finden Sie bei http://java.sun.com/, http://a-weinert.de/java, http://www.mozilla.org oder http://eclipse.org.

```
11.11.2004    45.635.523          jdk-1_5_0-doc.zip
11.11.2004    46.065.096          jdk-1_5_0-windows-i586.exe
11.11.2004    95.517.001  jdk-1_5_0-nb-4_0-beta2-bin-win.exe
```

Listing 1: Die Installationsdateien fürJDK1.5.0 (final)

Die größere der beiden Installationsdateien enthält neben dem JDK auch die SUN-Java-IDE NetBeans, die dann – wie im beispielhaften Vorgehen unten – gleich mit installiert wird. Die „kleinere“ Installationsdatei enthält lediglich das JDK. Das .zip-Archiv enthält die JDK-Dokumentation.

1.2 Beispielhafte Ausgangslage .

Für die Entwicklung und Verwendung von Java wird ein hinreichend ausgestatteter Pentium-Rechner mit Windows2000 verwendet (Bild 2). Installiert ist das JDK 1.4.2 im Verzeichnis

```
C:\programme\jdk
```

und der Programmsuchpfad des Betriebsystems zeigt (noch vor Windows-Anwendungen) auf

```
C:\programme\jdk\bin
```

(Listing 3).

Bild 2: Die Hardware- / Softwareplattform

```
D:\weinert\javaFactory> java -version

java version "1.4.2_05"
Java(TM) 2 Runtime Environment,
   Standard Edition (build 1.4.2_05-b04)
Java HotSpot(TM) Client VM (build 1.4.2_05-b04, mixed mode)

D:\weinert\javaFactory>path
PATH=C:\bat;c:\programme\util;C:\programme\jdk\bin;
     C:\WINNT\system32;C:\WINNT

D:\weinert\javaFactory>
```

Listing 3: Vorhandene JDK-Installation und Systemeinstellungen

Diese „versionsneutralen“ Installationsplätze und Pfadnamen und deren Hinterlegung in den Systemeinstellungen hat den Vorzug, dass jeweils neu installierte Java-Versionen allen direkten und indirekten Nutzern (z.B. auch Eclipse) ohne Zusatzmaßnahmen zur Verfügung stehen. Die funktionierende bisherige Installation rettet man durch vorheriges Umbenennen des Verzeichnisses

```
C:\programme\jdk
```

nach

```
C:\programme\jdk14
```

und ebenso einfach kann man zwischen mehreren Versionen umschalten (vgl. unten Listing 16).

Für die Software-Entwicklung werden Eclipse, CVS (Versionsverwaltung – holprig aber verbreitet) und (natürlich) die JDK-Werkzeuge (javac, javadoc, jar, jarsigner, keytool etc.) eingesetzt. Alle Erweiterungen und Tools, auf das sich das folgende explizit oder implizit bezieht, laufen mit 1.4.2 einwandfrei und alle betreffenden Quellen werden mit JDK1.4.2 ohne jede Errors oder Warnings übersetzt, “gejavadoct”, “gejart”, gejarsigned” etc., auch wenn dies im Folgenden nicht jedesmal erwähnt wird.

1.3 Zusätze, Erweiterungen .

Die JDK1.4.2-Installation (Ausgangslage) und deren Vorversionen werden unter anderem durch folgende Erweiterungen ergänzt:

- Framework
- einige Java-Anwendungen (utilities des Frameworks)
- Comm (Standardschnittstellen, parallel, seriell RS232)
- Xalan, Xerces (XML)
- JMX (java management extensions)
- JavaMail (eMail-Klassenbibliothek)
- Beans Activation Framework (u.a. nötig für mail)
- Ant (wenn überhaupt make, dann das)
- JDom (Java näheres XML als W3C-DOM)

Hinzu kommt jeweils, sowie soweit lieferbar oder aus zugänglichen Quellen erzeugbar, deren API-(javadoc-) Dokumentation. Listing 4 zeigt die wesentlichen Dateien.

```
22.12.2004      663.440            aWeinertBib.jar
07.03.2002       45.386            activation.jar
23.04.2003      737.884            ant.jar
30.01.2000       28.043            comm.jar
14.04.2003      135.381            jdom.jar
03.09.2003      365.858            jmxri.jar
03.09.2003      102.394            jmxtools.jar
07.03.2002      280.984            mail.jar
23.04.2003      671.546            optional.jar
16.06.2004    2.661.571       aWeinertBibDoc.zip
30.01.2000          467       javax.comm.properties
30.01.2000       27.648            win32com.dll
06.10.2003    2.952.605            xalan.jar
06.10.2003      895.924            xercesImpl.jar
06.10.2003      124.720            xml-apis.jar
```

Listing 4: Die Ergänzungsarchive zur Java-Installation

Diese Erweiterungen sind der jeweiligen JDK<=1.4.x-Installation jeweils passend mit entweder dem Mechanismus

- „installed extensions“

oder

- „endorsed standard“

hinzugefügt worden. Den alleinigen Einsatz dieser Mechanismen erkennt man an dem Symptom:

- „Niemals (nie!!) wird am „class-path“ herumgebastelt,
  weder als system property noch über Start-Parameter“

Dieses Prinzip ist bewährt und praktisch immer ratsam.

Die mit diesen Ergänzungen verbundenen Applikationen funktionieren nun auch noch mit Doppelklick auf .class- oder .jar-Files und mit der namentlichen Erwähnung des Klassennamens (ohne vorheriges java) in der Kommandozeile, wenn man die in Listing 5 angedeuteten Einstellungen vornimmt. Dies kommt dann Anwendern entgegen, die im Einzelfall gar nicht wissen wollen oder sollen, dass es sich um eine Java-Anwendung handelt.

Mit all dem (Listing 4) wollen wir nun nach einer Neuinstallation von JDK1.5.0 (+NetBeans) umziehen.

Zur nun geschilderten beispielhaften JDK1.4.x-Ausgangslage ist natürlich anzumerken, dass sich die Verhältnisse relativ leicht auf andere Verzeichnis- und Pfadeinstellungen und ein anderes Bündel von installierten Erweiterungen und Werkzeugen und mit ein wenig Mühe auch auf andere Betriebssysteme übertragen lassen, ohne dass das „dass und wie“ nun nochmals erwähnt wird.

```
@Echo Setting Java associations (W2K)
@echo (c) Copyright 2002, Albrecht Weinert
@Echo.
assoc .jar=jar-file
assoc .class=class-file
ftype jar-file=javaw -jar %%1 %%*
ftype class-file=runClass.bat  %%0 %%*
@Echo.
@Echo runClass.bat = start /i /d . /b (eine Zeile!)
@Echo        /wait java  %%~n1 %%~2
@echo.
@Echo Hinw.: set funktioniert nur lokal (ist nach Exit
weg). Machen Sie diese Einstellung mit der
Systemsteuerung permanent.
set PATHEXT=.COM;.EXE;.BAT;.CMD;.jar;.class
@Echo.
```

Listing 5: Windowseinstellungen für Java-Anwendungen

20 Java 1.5 + Zusätze in zehn Schritten .

2.1 Schritt 1: Die Grundinstallation des JDK1.5.0 .

Im bisher schon benutzten und gesetzten Standard-JDK-Installationsverzeichnis wird das neue JDK1.5.0 + NetBeans mit den folgenden Teilschritten installiert.

Die Tat

Teilschritt A) Umbenennen des Verzeichnisses

```
C:\programme\jdk
```

in

```
C:\Programme\jdk14
```

Wie bei Listing 3 beschrieben, zeigt der Programmsuchpfad des Betriebssystems auf ...\jdk\bin.

Teilschritt B) Neu Erzeugen von

```
C:\programme\jdk
```

Dieses neu erzeugte Verzeichnis muss leer bleiben. Der 1.5-Installer duldet im Gegensatz zu seinen Vorgängern keine "sitzen bleibenden" Dokumentationsverzeichnisbäume, installed extensions oder endorsed standards mehr.

Teilschritt C) Laufen Lassen des Installers

Für die Installation inklusive NetBeans startet man die aus Listing 1 (oben Seite 1) bekannte Datei

```
.....jdk-1_5_0-nb-4_0-beta2-bin-win.exe
```

Bild 6: JDK1.5.0- (+NetBeans-) Installation: Los geht's

Teilschritt D) Installationsverzeichnisse setzen

Nach bravem und üblicherweise blindem Klicken auf „Next" und „Yes" zu den selbstverständlich intensiv studierten Lizenzbedingungen muss man aufpassen, dass man das in Bild 7 gezeigte (dritte) Fenster nicht unbearbeitet durchlässt.

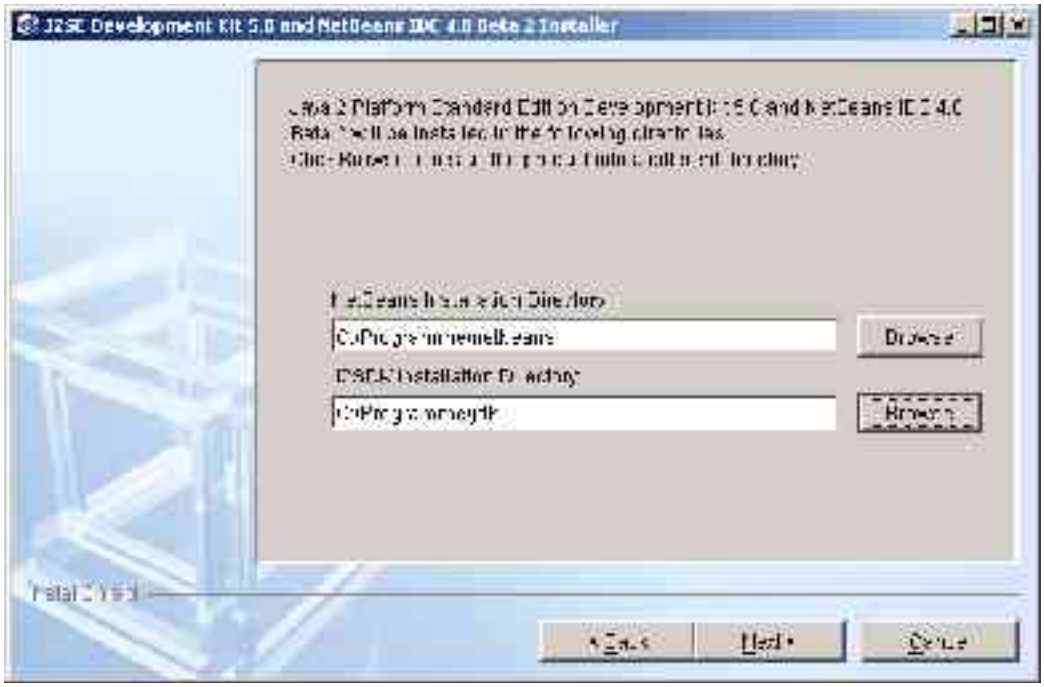

Bild 7: Einstellen der Pfade für die JDK1.5.0- (+NetBeans-) Installation

Hier stellt man nun den empfehlenswerten, „versionsneutralen" Installationspfad

```
C:\programme\jdk
```

ein. Ob man, wie in Bild 7 auch gezeigt, für NetBeans aus ähnlichen Gründen sinngemäß das selbe tut, ist Geschmacksache.

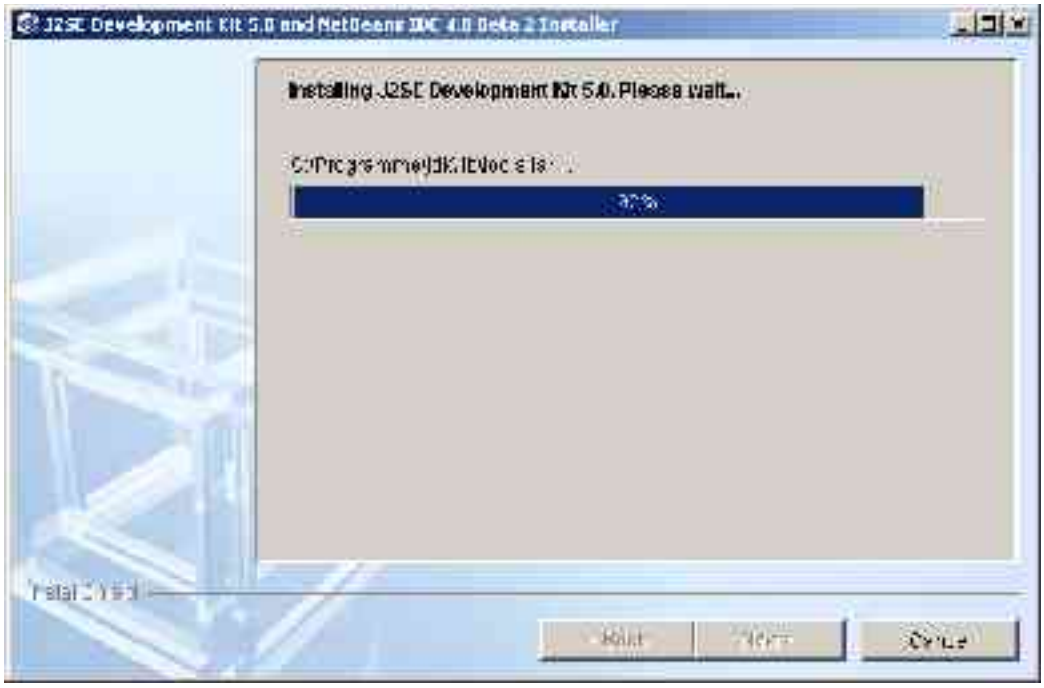

Bild 8: Die „heiße“ und längste Hauptphase der JDK1.5.0- (+NetBeans-) Installation

Teilschritt E) Abschließen der Installation

Durch nunmehr einfaches Weiterklicken mit „Next“ kommt man über die längere Hauptphase der Installation (Bild 8) zum mit „Finish“ zu bestätigenden Abschlussfenster (Bild 9).

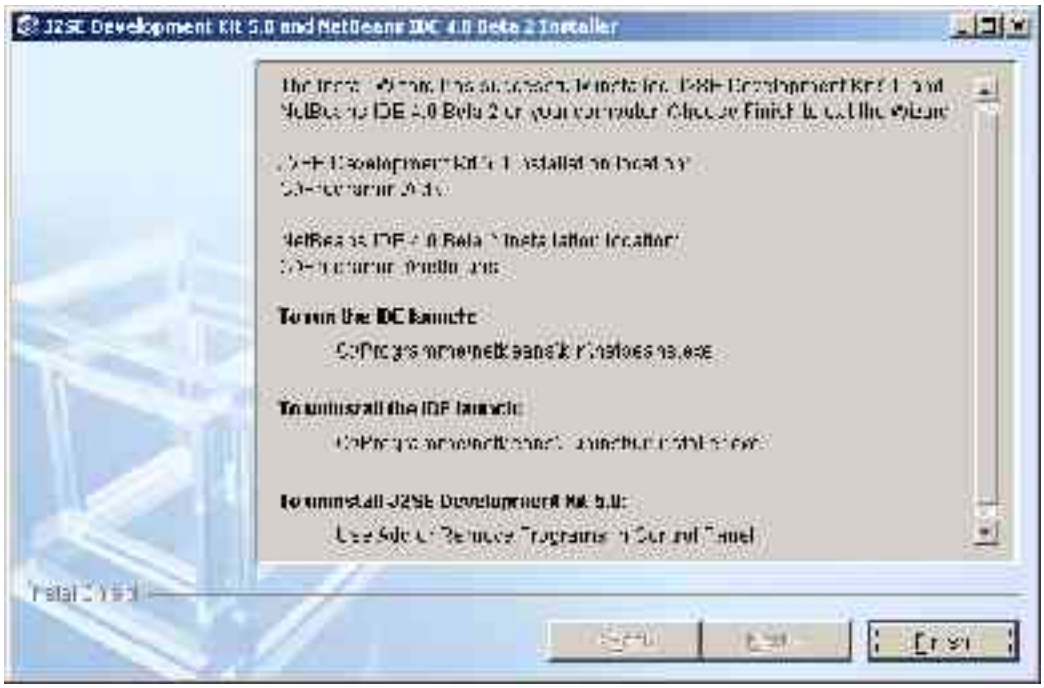

Bild 9: Abschlussbild der JDK1.5.0- (+NetBeans-) Installation

Anmerkungen

Es fällt positiv auf, dass die Handhabung der Installation recht einfach ist. Dass die Frage zwischendurch, ob man Beispiele, Quellen etc. auch haben wolle, fehlt, liegt daran, dass das mitinstallierte NetBeans das Quell-Archiv haben möchte (wie Eclipse ja übrigens auch). Bei der Installation ohne NetBeans muss man ein paar mehr Fragen beantworten.

Weiter fällt positiv auf, dass das im Gegensatz zu den Vorversionen weder das ganze JRE1.5.0 nochmals als Untermengenkopie installiert wurde noch die JVM-Startanwendungen java.exe und javaw.exe (zum bisher mindestens drittenmal) ins Windows-Verzeichnis ..\system32\ kopiert wurden. Letztere sollte und musste man ja am besten gleich „totschlagen“, aber dieser platzverschwendende und fehlerträchtige Unsinn hat ja nun endlich aufgehört.

Die Fehlerträchtigkeit liegt darin, dass man dabei dann eigentlich „installed extensions“ und „endorsed standards“ an drei Stellen installieren musste, von denen es bei einer (dem Windows-Verzeichnis) technisch gar nicht möglich war. Nur wer verzweifelte Anrufe des Typs „Java geht ja noch, aber ...“ liebt oder damit Geld verdient, wird diesen Fortschritt bedauern.

Auch diese Verbesserung gilt nicht bei der Installation ohne NetBeans. In diesem Fall muss als als Abschlussarbeit java.exe und javaw.exe aud den Windows-Systemverzeichnissen beseitigen.

Probe

Wenn alle Pfad-Einstellungen stimmig geblieben sind, bekommt man nun mit

```
java -version
```

von jedem Verzeichnis aus die Bestätigung, dass man Java 1 .5 hat; also Listing 10 statt Listing 3.

```
D:\weinert\javaFactory>  java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment,
    Standard Edition (build 1.5.0-b64)
Java HotSpot(TM) Client VM
    (build 1.5.0-b64, mixed mode, sharing)

D:\weinert\javaFactory>
```

Listing 10: Test neue (1.5) JDK-Installation

2.2 Schritt 2: Framework installieren

Tat

Das Java-Archiv des Frameworks wird von der bisherigen Java-Installation, deren Grundverzeichnis oben umbenannt wurde, in die neue Installation kopiert. Es wird also die Datei

```
aweinertbib.jar
```

von (umbenannt / gerettet)

```
C:\programme\jdk14\jre\lib\ext\
```

einfach nach

```
C:\programme\jdk\jre\lib\ext\
```

kopiert. (Ordentlicherweise holt man sich bei der Gelegenheit gleich die neuste Version.)

Probe

Man lässt eine einfache Hilfsanwendung dieses Frameworks, die sich garantiert auf keine weiteren ja noch nicht installierten Erweiterungen (siehe unten) abstützt, laufen. Dies muss von jedem Verzeichnis aus gehen.

Eine solche Anwendung ist ShowProps. Mit ihr kontrolliert man auch gleich mal die Systemeigenschaften der in Schritt 1 erfolgreich abgeschlossenen JDK-Installation; Listing 11 zeigt das Ergebnis.

```
D:\weinert\javaFactory>java ShowProps
 - - - - - -  Java - System - Properties - - - -    (51)

  ShowProps  V2.22 (Mo, 04.10.2004)

  Copyright (c)  1998 - 2001, 2004   Albrecht Weinert, Bochum


awt.toolkit               = sun.awt.windows.WToolkit
file.encoding             = Cp1252
file.encoding.pkg         = sun.io
file.separator            = \
java.awt.graphicsenv      = sun.awt.Win32GraphicsEnvironment
java.awt.printerjob       = sun.awt.windows.WPrinterJob
java.class.path           = .
java.class.version        = 49.0
java.endorsed.dirs        = C:\programme\jdk\jre\lib\endorsed
java.ext.dirs             = C:\programme\jdk\jre\lib\ext
java.home                 = C:\programme\jdk\jre
java.io.tmpdir            = C:\TMP\
java.library.path         = C:\programme\jdk\bin;.;
   C:\WINNT\system32;C:\WINNT;C:\bat; c:\programme\util;
   C:\programme\jdk\bin;C:\WINNT\system32;C:\WINNT
java.runtime.name         =
        Java(TM) 2 Runtime Environment, Standard Edition
java.runtime.version      = 1.5.0-b64
java.specification.name   = Java Platform API Specification
java.specification.vendor = Sun Microsystems Inc.
java.specification.version = 1.5
java.vendor               = Sun Microsystems Inc.
java.vendor.url           = http://java.sun.com/
java.vendor.url.bug       = http://java.sun.com/cgi-
bin/bugreport.cgi
java.version              = 1.5.0
java.vm.info              = mixed mode, sharing
java.vm.name              = Java HotSpot(TM) Client VM
java.vm.specification.name = Java Virtual Machine Specification
java.vm.specification.vendor = Sun Microsystems Inc.
java.vm.specification.version = 1.0
java.vm.vendor            = Sun Microsystems Inc.
java.vm.version           = 1.5.0-b64
line.separator            = (\n = ) 13 10
os.arch                   = x86
os.name                   = Windows 2000
os.version                = 5.0
path.separator            = ;
sun.arch.data.model       = 32
```

```
sun.boot.class.path       =
                        C:\programme\jdk\jre\lib\rt.jar;
                      C:\programme\jdk\jre\lib\i18n.jar;
                C:\programme\jdk\jre\lib\sunrsasign.jar;
                      C:\programme\jdk\jre\lib\jsse.jar;
                       C:\programme\jdk\jre\lib\jce.jar;
                  C:\programme\jdk\jre\lib\charsets.jar;
                  C:\programme\jdk\jre\classes
sun.boot.library.path     = C:\programme\jdk\jre\bin
sun.cpu.endian            = little
sun.cpu.isalist           = pentium_pro+mmx
                pentium_pro pentium+mmx pentium i486 i386 i86
sun.desktop               = windows
sun.io.unicode.encoding   = UnicodeLittle
sun.jnu.encoding          = Cp1252
sun.management.compiler   = HotSpot Client Compiler
sun.os.patch.level        = Service Pack 4
user.country              = DE
user.dir                  = D:\weinert\javaFactory
user.home                 = D:\weinert
user.language             = de
user.name                 = weinert
user.timezone             =
user.variant              =

D:\weinert\javaFactory>
```

Listing 11: Anzeige der Java-System-Eigenschaften mit ShowProps nach Framework-Installation (stark gekürzt; es sind insgesamt 51 Eigenschaften)

Anmerkungen

Dass unter 1.4 oder früher generierte installed extensions einschließlich darin enthaltener Anwendungen auch mit 1.5 laufen, mag bei den vielen Änderungen, die 1.5 brachte, zunächst erstaunen, und vielleicht hat bei Sun ja auch mancher dafür geschwitzt. Klar ist, dass es eigentlich auch gar nicht anders sein darf, denn sonst liefe ja vielfach nichts mehr.

Was man nun leider auch sieht, ist, dass die system property „user.timezone“ nach wie vor leer ist; probieren Sie es selbst. Vor langer Zeit ging das mal in genau einer Java-Version, und seitdem ist das ein „pending bug“.

2.3 Schritt 3: Comm-Extensions installieren

Die Installation der Comm-Extensions (javax.comm) beziehungsweise deren Kopieren von 1.4 nach 1.5 funktioniert wie obige Installation des Frameworks. Als kleine Komplikation gehören neben dem Java-Archiv noch zwei weitere Dateien dazu.

Tat

Teilschritt A) Das Archiv

```
comm.jar
```

wird vom entsprechenden Verzeichnis der bisherigen Installation (oder den entpackten Comm-Extensions) nach

```
C:\programme\jdk\jre\lib\ext\
```

kopiert.

Teilschritt B) Die ladbare Bibliothek

```
win32com.dll
```

wird nach

```
C:\programme\jdk\bin\
```

kopiert.

Teilschritt C) Die properties-Datei

```
javax.comm.properties
```

wird nach

```
C:\programme\jdk\jre\lib\
```

kopiert.

Probe

Man lässt eine einfache Hilfsanwendung des Frameworks, nämlich ShowPorts, die die Standard-schnittstellen des PCs zeigt, laufen. Sie würde bei nicht oder nicht korrekt installierten Comm-Extensions mit Meldung einer Exception (class not found) abbrechen.

Listing 12 zeigt das Ergebnis, das für andere PCs und Betriebssysteme natürlich anders aussehen wird (SerialA statt COM1 z.B.).

```
D:\weinert\javaFactory>  java ShowPorts

Port COM1 freie serielle Schnittstelle
9600 Baud, 8 Databits, 1 Stopbits, 0 par(kode)
 CD = false CTS = false DSR = false DTR = true RI =
false.

Port LPT1 freie parallele Schnittstelle
 PaperOut = false PrinterBusy = false
     PrinterError = false.

Port LPT2 freie parallele Schnittstelle
Port currently owned by Unknown Windows Application

D:\weinert\javaFactory>
```

Listing 12: Anzeige der Standard-Schnittstellen (nach comm-Extensions Installation)

Anmerkungen

Sun hat die Unterstützung der Standardschnittstellen immer noch nicht in das JRE/JDK einbezogen. Die Comm-Extensions werden auch nicht gepflegt. Ein Symptom ist der wirklich bejammernswerte Zustand der API-Dokumentation. Sie wurde mal mit einem ein halbes Duzend Versionen zurückliegenden JavaDoc erzeugt, sieht damit unpassend aus, hat keine Datei package.list und ist somit nicht mit der Dokumentation nutzender Klassen „verlinkbar“. Dies sind auch schon die einzigen schlechten Nachrichten in diesem Zusammenhang.

Diese uralten COM-Extensions für Windows liefen und laufen aber prima über alle Kombinationen von Windows-Varianten und JDK/JRE-Versionen. Prima heißt hier für eine serielle Schnittstelle unter Windows auch mit hohen Baudraten (38400, 57600, 115200) echtzeitgerecht. So kann man mit einer 100% pure Java-Anwendung mit Direktansteuerung des „Booster“ genannten Verstärkers Märklin-eisenbahn fahren (Treiber de.a_weinert.automation.SimpleMMBoosterDriver; Märklin-Motorola-

Protokoll). Etwas ernsthafter kann man auch mit einer jeweils im monatelangen Dauerbetrieb laufenden (ebenso 100% pure) Java-Automatisierungs-Anwendung mit unterlagerten als E/A-Konzentrator eingesetzten SPSen (Simatic) kommunizieren.

Ein weitere Java-Anwendung, die eine serielle Schnittstelle (via javax.comm) nutzt, ist ein industriell eingesetztes Test-Werkzeug für schlaue HART-Sensoren und -Aktoren. (HART ist ein auf analoge 4..20mA-Prozesssignale aufmoduliertes serielles Protokoll.)

Die Standardschnittstellen im PC sind ja schon öfter totgesagt worden. Viele „Kopplungs"-Anwendungen, wie die hier angedeuteten aus der Prozesssteuerung (einer in längeren Lebenszyklen denkenden Branche) mögen auch hier das Sprichwort, dass Totgesagte noch lange leben, bestätigen. Jeder, der (auch) über serielle Schnittstellen Prozessperipherie (im weitesten Sinne) ansteuern möchte, würde sich über etwas mehr Engagement von SUN bei der Unterstützung der Standardschnittstellen – in gleicher Qualität auf allen Plattformen – sehr freuen.

2.4 Schritt 4: JDK-Dokumentation installieren

Nun ist es an der Zeit, die Dokumentation des JDK (API, tools, guidelines etc.) zu installieren.

Tat

Teilschritt A) Ein Verzeichnis

```
C:\programme\jdk\docs
```

erzeugen; entweder mit dem Kommando „md" oder im Explorer.

Teilschritt B) Dieses Verzeichnis komprimieren;
entweder mit dem Kommando „compact" oder mit „Eigenschaften" im Explorer.

Bei Tausenden von .html-Dateien, aus denen die SUN- (javdoc-) Dokumentation (und natürlich auch eigene) besteht, bringt das "echt" Platz.

Teilschritte A und B als Kommandofolge:

```
D:\weinert\javaFactory>   cd /D C:\Programme\jdk
C:\Programme\jdk>         md docs
C:\Programme\jdk>         compact /C docs
```

Teilschritt C) Dokumentation auspacken.

Wechseln Sie in das Verzeichnis

```
C:\programme\jdk\
```

falls Sie sich nicht bereits (durch die gezeigte Kommandofolge) dort befinden.
Wechseln Sie auf **keinen** Fall in das gerade komprimierte (noch leere) Verzeichnis

```
C:\programme\jdk\docs  // hierher NICHT!
```

Packen Sie das Dokumentationsarchiv – das ist die .zip-Datei

```
jdk-1_5_0-doc.zip
```

aus dem Listing 1 – nun mit dem Kommando

```
C:\Programme\jdk> jar xfv  <woimmer>\jdk-1_5_0-doc.zip
```

aus. Dabei ist <woimmer> natürlich das Verzeichnis, in das Sie die 1.5-Installationsdateien runtergeladenen haben.

Probe

Mit dem Einstieg

```
file:///C:/Programme/jdk/docs/api/index.html
```

kann Ihr Mozilla nun die JDK1.5.0-API-Dokumentation lesen. Ein Verzeichnis höher werden Sie auch zu Werkzeugen, Richtlinien etc. geführt.

Anmerkungen

Das hier für das Auspacken der Dokumentation verwendete Werkzeug jar ist das Java-Archiv-Werkzeug. Es steht nach jeder erfolgreichen JDK-Installation und richtig gesetztem Anwendungssuchpfad (PATH) — siehe oben — überall zur Verfügung.

Dies ist damit auch eine Probe einer erfolgreichen JDK-Installation. Mit anderen Archiv-Werkzeugen (wie beispielsweise WinZip) bekommt man das Installieren der Java-Dokumentation auch hin, aber Java-Bordmittel reichen vollkommen aus.

Durch das gerade erfolgte Entpacken der JDK1.5.0-Dokumentation bekommen Sie in das Verzeichnis jdk\docs\ über 10.000 Dateien und über 700 Unterverzeichnisse. Die Gesamtgröße dieser Grund-Dokumentation ist 233 MB. Durch das beschriebene vorherige Komprimieren werden (von vornherein) aber "nur" 110 MB Platz beansprucht; das ist immerhin weniger als die Hälfte.

2.5 Schritt 5: Weitere Dokumentation installieren

In das Verzeichnis

```
....C:\programme\jdk\docs
```

wird nun auch (in jeweils passend benannte Unterverzeichnisse) die Dokumentation des Frameworks, der comm-Extensions sowie sonstiger weiterer Erweiterungen (javaMail, jDom und was sonst noch, s.u.) installiert bzw. diese bereits ausgepackte Dokumentation von der vorherigen (geretteten / umbenannten) Installation kopiert.

Tat

Wenn Ihnen die betreffende Dokumentation in Archiv-Form vorliegt, gehen Sie jeweils sinngemäß wie im Teilschritt C der vorangegangenen Schritts vor. Falls jedoch vorhanden, kopieren Sie einfach die jeweils betreffenden Verzeichnisse (mit allen Unterverzeichnissen) von

```
....C:\programme\jdk14\docs
```

nach

```
....C:\programme\jdk\docs
```

Konkret wären dies die Verzeichnisse aWeinertBib, commapidocs sowie für die unten folgenden Erweiterungen die Verzeichnisse jdom und mailapidocs, jeweils mit allen Unterverzeichnissen.

Probe

Gehen Sie in der eben kopierten oder ausgepackten Framework-Dokumentation testweise nach

```
file:///C:/Programme/jdk/docs/aWeinertBib/de/a_weinert/Prop.html
```

Sie sehen die Dokumentation dieser Klasse (Prop). Klicken Sie nun dort auf AbstractMap (als Link zu finden u.A. ganz am Anfang bei der Darstellung der Erbfolge). Wenn alles korrekt installiert und verzeichnisrichtig platziert wurde, sind Sie nun in der im vorigen Schritt installierten 1.5.0-SUN-API-Dokumentation der Klasse

```
java.util.AbstractMap<K,V>
```

Anmerkungen

Das Kopieren der jeweiligen (JavaDoc-erzeugten) Dokumentationsdateibäume in die neue Installation anstelle von neu Erzeugen oder Auspacken aus neu zu ladenden Archiven ist möglich, da es bei diesen Klassendateien und Erweiterungen (noch) keine Änderungen beim Schritt nach 1.5. gab. Wie weit das passt, ist ja ein Gegenstand dieser Untersuchung.

Wenn man aber nun bei einem Schritt zur nächsten JDK-Version für weiterverwendete Frameworks und Klassenbibliotheken die vorhandenen Dokumentationsdateibäume eh‘ unverändert weiterverwenden kann, warum soll man sie dann, wie in diesem Schritt beschrieben, in die neue JDK-Installation kopieren bzw. dort noch mal genauso auspacken? Dem Mozilla (und anderen .html-Betrachtern) ist es ja schließlich egal, wo ein in sich relativ verlinkter Dateibaum absolut steht.

Der Grund sind die relativen Links zur Dokumentation anderer Bibliotheken und vor allem zur JDK-Dokumentation. Der zweite Teilschritt der Probe würde ohne die Kopie des Dokumentationsbaums in die neue Installation misslingen. Genauer gesagt würde man mit dem vorgeschlagenen Klick statt bei der nun ja verwendeten JDK1.5.0-AbstractMap<K,V> bei deren nun ja doch deutlich anderen JDK1.4.2-Vorgängerin landen.

Also alle zwar getrennt mit JavaDoc generierten aber untereinander relativ verlinkten Dokumentationsbäume von Klassenbibliotheken müssen im selben Grundverzeichnis „wohnen“. Und die naheliegende Wohnortswahl ist das Verzeichnis docs\ der (aktuellen) JDK-Installation.

Die beispielhafte Kommandofolge

```
PushD C:\programme\jdk\docs\aWeinertBib
del /s /q C:\programme\jdk\docs\docs\aWeinertBib\*.*
Javadoc -author -version -protected -use -source 1.4  -J-mx32m
  -J-ms32m  -link ../api  -link ../mailapidocs  -link ../jDOM
     -sourcepath D:\Weinert\javaFactory\
        @D:\Weinert\javaFactory\DEawDoc.list
PopD
```

mag als Anregung dienen, wie man eine solche relativ mit anderen verlinkte Dokumentation neu erstellt. (Alle Kommandos kommen in jeweils einer Zeile. Die – lange – Liste der zu dokumentierenden Pakete und Einzelquellen ist hier in die Datei DEawDoc.list ausgelagert.)

Wer als Leser einer JavaDoc-Dokumentation einmal die Arbeitserleichterung durch solche relativen Links kennen gelernt hat, möchte diesen Komfort nie wieder missen. Dass dieses JavaDoc-Feature – sprich die -link-Option – von Entwicklern bedauerlicherweise eher selten genutzt wird, mag daran liegen, dass das guten Gelingen eine etwas kniffelige Handhabung von Arbeitsverzeichnissen und Verweisen erfordert, wie die obige Beispiel-Kommandofolge auch zeigt. (Gemeint ist nicht das leider gar nicht so kleine „Entwickler-Subset“, das eh‘ keine javaDoc-Dokumentation schreibt.)

2.6 Schritt 6: Übrige Erweiterungen – mail, Ant

Sie haben auf der Java-mail-API beruhende vorhandene Anwendungen, wie z.B. SendMail (im o.g. installierten Framework) oder eine Automatisierungsanwendung, die beim Eintritt in bestimmte (Alarm-) Zustände e-Mails (mit SMTP) senden will. Damit dies auch unter 1.5.0 läuft, müssen Sie die Java-mail-API (javax.mail) und folglich auch das Beans-Activation-Framework installieren.

Und vielleicht nutzen Sie auch das bessere make-Tool „Ant“.

Tat

Teilschritt A) mail

Die Dateien (vgl. Listing 4)

```
mail.jar
```

und

```
activation.jar
```

werden (von der vorangegangenen Installation) in das Verzeichnis

```
C:\programme\jdk\jre\lib\ext\
```

kopiert.

Teilschritt B) Ant

Für ggf. Ant tun Sie dasselbe mit den Dateien

```
ant.jar
```

und

```
optional.jar
```

```
<project name="TestSound" default="sound" basedir=".">
  <target name="sound"><sound>
    <success  source="logon.wav" />
    <fail  source="logon.wav" />
  </sound></target>
</project>
```

Listing 13: Kleines Ant -Test-„Projekt“ (für optional.jar)

Probe

Einfache Tests mit Ant-Projektdateien zeigen, dass das meiste geht. Mit der verbose-Option (-v) behauptet Ant hartnäckig ein Java 1.4 in einem Verzeichnis gefunden zu haben, in dem sich nun garantiert keines befindet. Was (zumindest so einfach ohne weiteres) auch nicht geht, ist die Tonausgabe (mit Listing 13), die unter 1.4 (auch nur mit optional.jar) problemlos geht.

```
Subject: Test SMTP mit Java mail API
Date: Sun, 11 Nov 2004 12:18:31 +0100
To: "Höchstselbst" <ich@meiner.domain>

Dies ist ein Test.
```

Listing 14: Kleine e-Mail-Definitions-Testdatei nach RFC 822 für SMTP
(Die Leerzeile ist wesentlich.)

Das Senden einer e-Mail, die einer in einer Textdatei, wie in Listing 14 gezeigt, gemäß RFC822 definiert ist, mit dem Kommando

```
java SendMail -v -parse mailDefNachRFC822.txt
```

geht nun genau so wie unter <= 1.4 auch unter 1.5.

Um es deutlich zu sagen, wenn es vorher nicht ging, weil z.B. ihr PC keinen Zugriff auf einen SMPT-Server hat, oder Sie schon unter 1.4. nicht die dazu passenden Einstellungen in den entsprechenden . properties-Dateien hatten, brauchen Sie es unter 1.5 nicht erst zu versuchen.

Anmerkungen

Hier hätte man erwarten können, dass die mail-API mit allem Drum und Dran ab 1.5. zum Standard gehören würde. Aber mit dem einfachen Nachinstallieren der beiden Extensions (und möglichst auch deren Dokumentation; s.o.) geht’s ja auch.

Dass der Ant-Test mit Listing 13 fehlschlägt, liegt vermutlich daran, dass die Ant-Entwickler (wie schon andere aus purer Verzweiflung) für das Abspielen einer .wav-Datei wohl auf inoffizielle .sun-Pakete zugreifen, die beim Schritt von 1.4 nach 1.5 natürlich nicht mehr so weiterfunktionieren müssen. Es wird einfach Zeit, dass Sun für das einfache Abspielen eines Sounds aus einer Datei bzw. einem Stream (plus Nichts) durch eine Applikation einen einfachen robusten und unaufwändigen Weg (ohne midis und Soundbanks etc.) offiziell bereitstellt. Anders formuliert: Was man in der Basis-Plattform mit einem Klick zum „Quaken“ bringt, muss aus einer Java-Applikation mit einer Zeile abzuspielen sein.

2.7 Schritt 7: Übrige Erweiterungen – JMX .

Sie haben auf JMX (Java management extensions) beruhende vorhandene Anwendungen. Diese nutzen, wie das Framework, auch die (bisherige) Referenz-Implementierung des JMX-HTML protocol adapter. Nach der etwas zu einschränkenden Betrachtungsweise Vieler macht diese http-Fern-BuB (Bedienung und Beobachtung) von Anwendungen das ganze JMX ja erst liebenswert.

Nun gehört JMX (an sich) seit 1.5 zum JDK/JRE-Standardumfang, nicht aber eine HTML protocol adapter Implementierung.

Tat

Wenn Sie JMX gar nicht oder ohne den HTML protocol adapter.verwenden tun Sie ab 1.5.0 nichts mehr. Ansonsten installiert man von der bisherigen umfangreicheren JMX-Nachinstallation (einschließlich Dokumentation!) nur noch die Datei

```
jmxtools.jar
```

nicht aber andere nach JMX riechende Dateien (vgl. Listing 4) nach

```
C:\programme\jdk\jre\lib\ext\
```

Probe

Ein Test mit JMX-Anwendungen zeigt, dass JMX nun wie vorher, also einschließlich http-BuB, funktioniert.

Anmerkungen

Die Übernahme von JMX in den Standard JDK/JRE-Lieferumfang und damit auf mittlere Sicht deren Vorhandensein auf jeder Java-Kundenplattform kann die Verbreitung dieser nützlichen Technologie nur fördern.

2.8 Schritt 8: Übrige Erweiterungen – XML mit und ohne JDom

Die XML-Unterstützung im Standardfunktionsumfang wurde mit 1.5 deutlich erweitert und verbessert. Was man, beispielsweise wegen Schema-Validierung, an „Xalanen“ und „Xercenen“ (Namen von XML-Java-Implementierungen) als endorsed standard nachinstalliert hatte, ist nun i.A. nicht mehr nötig. Dies gilt nicht für jDOM.

Tat

Wenn bei Ihrer bisherigen Installation im Verzeichnis

```
.....jdk14\jre\lib\endorsed\
```

also beispielsweise die Dateien

```
xalan.jar, xercesImpl.jar und xml-apis.jar
```

standen, so lassen Sie das Verzeichnis

```
C:\programme\jdk\jre\lib\endorsed\
```

nun (diesbezüglich) leer bzw. erzeugen Sie es gar nicht erst.

Falls Sie jDOM als installed extensions hatten (und es nach wie vor brauchen), so kopieren Sie von Ihren bisherigen XML-Erweiterungen die Datei

```
jdom.jar
```

(und nur diese) in das Verzeichnis

```
C:\programme\jdk\jre\lib\ext\
```

und kopieren Sie dann auch die vorhandene jDOM-Dokumentation an ihren neuen Platz (s.o.).

Probe

Das Laufen lassen einiger einfacher XML-Anwendungen (Validierungen, Transformationen) ggf. auch jDOM-basierter, zeigt keine Probleme.

Anmerkungen

jDOM ist ein an sich vernünftiger Java- und OO-näherer Ansatz für eine Java-XML-Klassenbibliothek als das W3C-DOM. jDOMs ursprüngliche Chancen, mal JDK/JRE-Standard zu werden, scheinen ja eher zu sinken. Die erneut verbesserte integrierte XML-Unterstützung ohne Einbeziehung von jDOM weist in diese Richtung und wirft (erneut) die Frage auf, ob man ggf. vorhandene jDOM-basierte Anwendungen nicht zum nun deutlich verbesserten Java-XML-Standard zurückportieren sollte.

Andererseits ist, wie gesehen, die Beibehaltung der jDOM-Erweiterung auch unter 1.5.0 einfach möglich.

2.9 Schritt 9: Erster Test von NetBeans .

Wer im ersten Schritt nicht die „kleine“ JDK-Installationsdatei genommen hat, hat ja ganz „nebenbei“ die Java-IDE NetBeans mit installiert. Ein Doppelklick auf die Ikone startet (als allererste Probe) diese IDE auch klaglos, wenn auch relativ zu Eclipse noch langsamer.

Dass die üblichen Beispielprojekte und „Mit drei Mausklicks zu einem CVJM-Service“-Demos funktionieren (und wenig beweisen) vollzieht man schnell nach oder glaubt es auch so. Ein bisheriger Eclipse-Nutzer und NetBeans-Neuling möchte mit einem vorhandenen größeren Projekt wissen, wie weit man mit der neuen IDE kommt.

Tat

Teilschritt A) Testprojekt in Quarantäne-Verzeichnis aufsetzen.

Kopieren Sie ein mittelgroßen Projekts (Duzend Pakete, Hunderte Klassen) von Eclipse in ein temporäres Test-Verzeichnis. Machen also etwas entsprechendes wie das Kopieren von

```
....Eclipse\workSpace\weBib
```

mit allen Unterverzeichnissen nach

```
C:\Temp\TestnetBeansProject\weBib
```

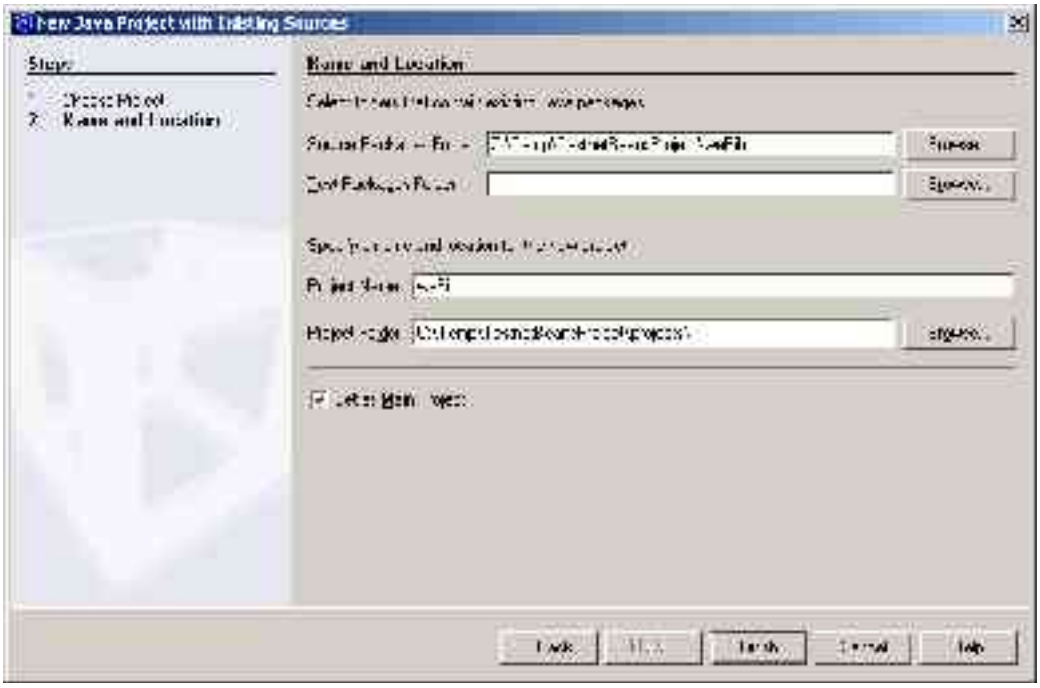

Bild 15: Einrichten des (aus Eclipse kopierten) NetBeans-Test-Projekts

Durch ein solches Test- bzw. Quarantäneverzeichnis kann man nun beliebig herumprobieren, ohne was kaputtzumachen.

Teilschritt B) NetBeans Projekt mit vorhandenem einrichten

In NetBeans richten Sie ein "New Java Project with existing sources" mit diesem Verzeichnis als Quellverzeichnis ein; siehe Bild 15.

Probe

Nehmen Sie nun in NetBeans Ihre gewünschten Einstellungen zu Einrückungen etc. (ähnlich wie bei Eclipse) vor und arbeiten Sie mit dem Projekt. Ein vorher unter Eclipse und 1.4 fehlerfrei „baubares“ Projekt lässt sich auch unter 1.5 und NetBeans bearbeiten und fehlerfrei nachbauen.

Hinzu kommen (bei Nicht-Nutzung der entsprechenden 1.5-Möglichkeiten) lediglich Warnungen wegen „nicht generischer“ und als „unsafe“ titulierter Verwendung von Containern aus dem collection framework. Diese Warnungen bekommt man auch bei direktem Aufruf des 1.5-Java-Compilers.

Anmerkungen

Mit NetBeans (4.0 beta2 mit 1.5.0) scheint einiges langsamer und zäher zu laufen als Eclipse (3.0 mit 1.4.2). Auch scheinen einige „Syntax unterstützende“ Features, die vielleicht auch Eclipse den Umstieg nach 1.5 so problematisch machen, einfach zu fehlen. Ansonsten ist es eine tolle Sache, mit der JDK-Installation fast unbemerkt eine leistungsfähige und passende IDE mit zu bekommen.

Ein zweigleisiges Fahren bei den IDEs mag in nächster Zeit durchaus angesagt sein; bei 1.5-Sprachelementen, Webservices, J2EE etc. eher NetBeans und beim 1.4-Sprachumfang Aplikationen und Bibliotheken eher Eclipse mit fließenden Übergängen.

2.10 Schritt 10: Erster Test der wesentlichen JDK1.5.0-Werkzeuge

Tat

Um unter 1.5 Eclipse und die wesentlichen JDK-Werkzeuge – javac, jar, javadoc, jarsigner – zu testen muss nichts mehr installiert werden.

Probe

Ein (bereits mehrfach angesprochenes) mittelgroßes Projekt wird mit Eclipse betrachtet und (außerhalb) Eclipse mit Steuerdateien komplett neu gebaut.

Zwei Dinge funktionieren nicht: Eclipse und javaDoc.

Anmerkungen

Wenn man Eclipse (3.0), wie hier geschehen (quasi „heimlich“) eine andere Java-Installation unterschiebt, geschieht dies ja gleich in zwei Rollen: einmal als Laufzeitbasis für Eclipse selber, das ja eine Java-Anwendung ist, und zum zweiten als Entwicklungsbasis für die bearbeiteten Projekte. Für jede Java2-Version kann man dies vorwärts und rückwärts tun, und Eclipse muckt nicht. Dass dies für den Schritt nach 1.5 und ggf. wieder zurück nicht so einfach sein würde, war abzusehen. (Ja es gibt Lösungen. Diese sind deutlich komplizierter in der Handhabung, und sie trennen i.A. die beiden genannten Rollen. Ein solches Vorgehen passt nicht zur vorliegenden Fragestellung).

Dass javaDoc (außer für Einfachstfälle) versagt, ist unerwartet. Es ist deutlich ärgerlicher, zumal es die Wiederholung eines schon mal gemachten Fehlers (ein sogenannter „regression bug“) ist. Aber Sun arbeitet dran (Bug ID: 5101868: „javadoc crashes ... with NullPointerException“), und so wird 1.5.1 wohl auch hierbei fehlerfrei sein.

Der javaDoc-Fehler lässt das automatische Durchgenerieren von Projekten „platzen“. Dies und die angedeuteten jeweiligen Einschränkungen von Eclipse und NetBeans, wird wohl dazu führen, dass man noch eine ganze Zeit nicht nur für Tests sondern auch während der Entwicklung zwischen JDK1.4.x und JDK1.5.x hin und her wechseln wird. Listing 16 zeigt einen bewährten (während der Vorbereitung des Vorliegenden bestimmt hundert mal benutzten) Weg dazu als Anregung.

```
@Echo.
@Echo SwitchJava V01.01 12.07.2004 (c) Albrecht Weinert
@Echo.
@if not exist C:\Programme\jdk goto :alrS
@Echo Jetzige / bisherige  JDK - Version:
@C:\Programme\jdk\bin\java.exe -version
@Echo.
@if %1X==X goto :aufruf

@if %1==15   goto :j15
@if %1==-15  goto :j15
@if %1==1.5  goto :j15
@if %1==-1.5 goto :j15
@if %1==14   goto :j14
@if %1==-14  goto :j14
@if %1==1.4  goto :j14
@if %1==-1.4 goto :j14
:aufruf
@Echo Aufruf: SwitchJava 1.4 oder SwitchJava 1.5
@goto :end

:j15
@Echo Switch to 1.5
if not exist C:\Programme\jdk15 goto :alrS
if exist     C:\Programme\jdk14 goto :alrS

rename C:\Programme\jdk    jdk14
@if ERRORLEVEL 1 goto :Error
rename C:\Programme\jdk15  jdk
@if ERRORLEVEL 1 goto :Error
@goto :end

:j14
@Echo Switch to 1.4
if exist C:\Programme\jdk15 goto :alrS
if not exist C:\Programme\jdk14 goto :alrS

rename C:\Programme\jdk jdk15
@if ERRORLEVEL 1 goto :Error
rename C:\Programme\jdk14 jdk
@if ERRORLEVEL 1 goto :Error

:end
@Echo.
@Echo Aktuelle / Neue  JDK - Version:
@C:\Programme\jdk\bin\java.exe -version
goto :stopp

:error
@Echo.
@Echo ERROR
:alrS
@Echo.
@Echo Bereits gesetzt oder Directory-(Zugriff-) Probleme.

:stopp
@Echo.
```

Listing 16: Batch-Datei SwitchJava.bat zum Umschalten zwischen JDK 1.4 und 1.5

3 Ergebnisse, Ausblick

Der große Schritt, den die Sprache Java von JDK1.4.2 nach 1.5.0 gemacht hat, wird bei Ausnutzung der neuen syntaktischen Möglichkeiten oder der Erweiterungen der Klassenbibliotheken (java.lang.StringBuilder z.B.) zu einer „journey of no return“ – und angesichts der zahlreichen Verbesserungen auch eine Reise ohne Bedauern. Hier wurde die Frage untersucht, wieweit dieser gewaltige Schritt – ähnlich wie bisherige „minor changes” von JDKx.y.z nach JDKx.y.(z+1) – sich mit vorhandenen Projekten, Werkzeugen und Erweiterungen verträgt. Im Lichte dieser Fragestellung zeigt JDK1.5.0 trotz der Änderungen in einem wohl noch nie da gewesenen Umfang eine sehr erstaunliche Rückwärtskompatibilität und weitgehende Problemlosigkeit.

4 Appendix

4.1 Autor

Bild 17: Weinert

Professor Dr.-Ing. Albrecht Weinert ist Jahrgang 1953. Nach dem Studium der Physik an der TU Clausthal promovierte er am dortigen Institut für Elektrische Energietechnik mit einer Dissertation über die computer-gestützte Modellierung des thermischen Verhaltens sehr großer Gleichstrommaschinen zum Dr.-Ing.

Von 1984 bis 1997 arbeitete er in der Entwicklung im Bereich Automatisierungstechnik der Siemens AG in Karlsruhe auf den Gebieten der Redundanz, Fehlertoleranz und Sicherheit in der Prozessleittechnik. Er war maßgeblich an der Architektur und Systementwicklung bei TELEPERM M und SIMATIC S7 beteiligt.

Seit März 1997 ist der Autor Professor an der Fachhochschule Bochum im Fachbereich Elektrotechnik und Informatik. Der Autor des des Lehrbuchs Java für Ingenieure hat schon früh (Wintersemester 1997) die Sprache Java an der Fachhochschule Bochum eingeführt und damit den Fachbereich und sein Labor für Medien und verteilte Anwendungen, kurz MEVA-Lab, an die Spitze einer bis heute ungebrochenen Entwicklung gestellt.

Link: http://www.a-weinert.de

4.2 Literatur

Java

[JavaWe] Weinert, Albrecht, Java für Ingenieure, Hanser, München, Leipzig, 2002

[JavaPL] Arnold, Ken, & Gosling, James, The Java™ Programming Language, The Java Series from Addison Wesley, 4. Auflage (oder höher)

[JavaLSp] Gosling, James, Joy, Bill, & Steele, Guy, The Java™ Language Specification, The Java Series from Addison Wesley, 4. Auflage (oder höher)

[Riehle96] Gamma, Erich, Helm, Richard, Johnson, Ralph & Vlissides, John, ins Deutsche übersetzt von Riehle, Dirk, Entwurfsmuster (Originaltitel: Design Patterns), Addison-Wesley, Deutschland, 1996

[CvsCed] Cederqvist, Per, Version Management with CVS (for cvs 1.11.3), WWW

[RcsTichy] Tichy, Walter F. Design, Implementation, and Evaluation of a Revision Control System, *Proceedings of the 6th International Conference on Software Engineering*, IEEE, Tokyo, September 1982.

[Bail2000] Bailliez, Stephane, et alii, Ant User manual, Apache Software Foundation 2000, 2001

[McL2000] McLaughlin, Brett, Java and XML, O’Reilly 2000

[OTI2003] Object Technology International, Inc, Eclipse Platform Technical Overview, OTI 2003

4.3 Abkürzungen .

A

ANSI	American National Standards Institute
API	Programmierschnittstelle zu allgemein verwendbaren Bibliotheken und den Funktionen des Ziel-Betriebssystems (Application Programming Interface)
ASCII	American Standard Code for Information Interchange, Festlegungen für die Kodierung von (Text-) Zeichen
AWT	Abstract Window Toolkit, plattformunabhängige Programmierschnittstelle für grafische Ein- und Ausgabefunktionen

B

B&B	Bedienen und Beobachten von Prozessen (auch BuB)

D

DDL	Data definition language
DLL	Dynamic Link Library; zur Programmlaufzeit ladbare Bibliothek
DTD	Document Type Definition.

E

E/A	Ein- und Ausgabe

F

FAQ	Frequently Asked Questions (Hilfetexte in Frage-Antwort-Form)
FH	Fachhochschule

H

HTML	Hypertext Markup Language. Beschreibungssprache für verknüpfte Seiten, die unter anderem mit HTTP im WWW übertragen werden (RFC 1866).
HTTP	Hypertext Transfer Protokoll. Ein Internet-Protokoll zur Übertragung von WWW-Seiten.

I

IDE	Integrated development environment; integrierte Entwicklungsumgebung
IDL	Interface Definition Language (insbesondere der OMG)
I/O	Ein- / Ausgabe (von Prozesssignalen; input/output)

J

J2EE	Java 2 Enterprise Edition
JAF	JavaBeans Activation Framework
JAR	Java-Archiv. Insbesondere zur Zusammenfassung mehrerer zu einer Anwendung oder einem Applet gehörender Klassen- und sonstigen Dateien. Das Dateiformat entspricht .zip. JAR ist auch das JDK-Werkzeug zum Erstellen und Handhaben solcher Archive.
JDC	Java Developer Connection (Ein WWW-Service)
JDK	Java Development Kit; der Werkzeugsatz für die Entwicklung mit Java
JEB	Enterprise JavaBeans (ungleich JavaBeans)
JFC	Java Foundation Classes (grafische Klassenbibliothek = Swing)
JIT	Just in time compiler Ergänzung eines Emulators um Übersetzungsfunktionen mit dem Ziel höherer Leistung
JNDI	Java Naming and Directory services Interface
JNI	Java Native Interface

JRE	Java Runtime Environment; JDK-Subset ohne Entwicklungswerkzeuge, reine Laufzeitumgebung
JSP	Java Server Pages
JSSE	Java Secure Socket Extension (seit JDK1.4.x integriert)
JVM	Java virtual machine; der eigens für Java erfundene Prozessor
N	
NT	New Technology (meist synonym für MS-Windows NT)
NTFS	Windows NT File System
O	
ODMG	Object Database Management Group
OO	Objektorientierung, objektorientiert
OOP	Objektorientiertes Programmieren
P	
PLC	Programmable Logic Controller (umfasst u.A. SPS)
POP	Post Office Protocol (i.a. mit nachgestellter Versionsnummer: POP3)
R	
RFC	request for comment; Internet-Standards und -Vornormen
RMI	Remote Method Invocation; Aufruf einer Methode auf einem anderen Knoten
RTTI	Run time type information; Laufzeittypinformation
S	
STL	Standard Template Library, Bibliothek mit generischen Einheiten
SMTP	Simple Mail Transfer Protocol (ein TCP/IP-Protokoll für E-Mail)
SPS	Speicherprogrammierbare Steuerung, kleines Automatisierungsgerät
STL	Standard Template Library, Bibliothek mit generischen Einheiten
U	
UML	Unified Modelling Language; Sprache zur Darstellung von Objektbeziehungen
URL	Uniform resource locator; Adresse einer Datei im Internet
USART	universelle, serielle, asynchrone Schnittstelle zum Empfangen und Senden
V	
V.24	Serielles Übertragungsprotokoll
VM	Virtual Machine, gedachte oder emulierte Rechnerarchitektur
W	
W3	Amerikanische Kurzform für WWW
W3C	WWW-Consortium ; Standardisierung der WWW- Verfahren und Protokolle, RFCs
WWW	World Wide We b, Gesamtheit der HTML-Seiten im Internet
X	
X86	INTEL-80x86-Rechnerarchitektur, binärkompatible Linie von 8086 über 80286, 80386, 80486 bis Pentium.
XML	Extensible mark-up language, erweiterbare Datenbeschreibungssprache