Java in der Ausbildung

Robert Tolksdorf[1] und Wilhelm Weisweber[2]

[1] Technische Universität Berlin, Fachbereich Informatik, FLP/KIT,
Sekr. FR 6–10, Franklinstr. 28/29, D-10587 Berlin, Germany,
mailto:tolk@cs.tu-berlin.de, *http://www.cs.tu-berlin.de/~tolk/*
[2] *mailto:ww@cs.tu-berlin.de*, *http://www.cs.tu-berlin.de/~ww/*

Zusammenfassung Java spielt in der Informatik Aus-, Fort- und Weiterbildung eine immer stärkere Rolle. Dieses Papier faßt die positiven Erfahrungen der Autoren mit dem Einsatz von Java in der Lehre zusammen. Die Hauptvorteile der Sprache für die Ausbildung liegen in der Modularisierbarkeit von Lerninhalten, der Skalierbarkeit der Sprache über Anforderungsniveaus und ihrer Relevanz für die Berufspraxis. Nachteile wie schlechte Lehrbuchsituation oder unintuitive Entwicklungsumgebungen stehen Entscheidung für Java in der Ausbildung keineswegs entgegen.

1 Einleitung

Die Programmiersprache Java hat in den letzten Jahren das Internet verändert und objektorientierte Programmierung popularisiert. Die starke Verbreitung und einfache Verfügbarkeit, der gute Standardisierungsgrad und die hohe Qualität des Sprachentwurfs machen Java für den Einsatz in der Aus-, Fort- und Weiterbildung in Informatik attraktiv.

Obwohl Java in nennenswertem Ausmaß in der Lehre objektorientierter Programmierung eingesetzt wird, haben sich Java wie auch andere objektorientierte Programmiersprachen noch nicht in der Informatikausbildung etabliert. Die meisten Berichte über die Verwendung in der Ausbildung kommen aus den USA. In Deutschland gibt es vergleichsweise wenig Erfahrungsberichte. Im *Informatik Spektrum 20(6)* vom Dezember 1997 war die Objektorientierung in der Ausbildung das Hauptthema, was das steigende Interesse an diesem Thema andeutet.

Leider lassen sich Ergebnisse aus dem englischsprachigen Raum nicht ohne weiteres auf Deutschland übertragen, weil die ersten Studienjahre dort inhaltlich mehr standardisiert sind. An deutschen Universitäten sind die Inhalte im Grundstudium gerade im Hinblick auf die Verwendung von Programmiersprachen heterogener. Es ist daher wichtig, möglichst viele Erfahrungsberichte über die Ausbildung mit objektorientierten Programmiersprachen an Hochschulen und in der Industrie zu erstellen, um aussagefähige Ergebnisse zu bekommen.

Die Autoren verwenden Java als Gegenstand und als Mittel in der Informatikausbildung mit Bezug zur Objektorientierung an Hochschulen, in öffentlichen Verwaltungen und in der Industrie:

- *Informatikausbildung* in Kursen, Praktika und Projekten in objektorientierter Programmierung im Grund- und Hauptstudium der Informatik an der TU Berlin und der Univ. Hildesheim.

  - Grundstudium: Praktikum OO Programmierung, Einführung in Java[1]
  - Hauptstudium: Basis von Programmiersprachen und -systemen
  - Hauptstudium: Projekt Entwicklung komplexer Anwendungssysteme
- *Informatikweiterbildung* mit Kursen an der Verwaltungsakademie Berlin, die sich an Mitarbeiter des öffentlichen Dienstes wenden.
- *Informatikfortbildung* für Entwicklerteams in Firmen.

Mit diesem Papier möchten die Autoren ihre Erfahrungen weitergeben und kritisch reflektieren. Es soll Lehrende bei der Auswahl einer Programmiersprache für die Informatikausbildung unterstützen. Außerdem wird versucht, die didaktische und inhaltliche Eignung von Java nachzuweisen. Insgesamt ist dieses Papier ein Plädoyer für die Verwendung von Java in allen Ausbildungsabschnitten.

Dieses Papier enthält im folgenden Abschnitt eine Bestandsaufnahme der Erfahrungen mit der Verwendung von Java in der Ausbildung. Der dritte Abschnitt geht auf die Kriterien zur Auswahl einer Sprache für die Ausbildung ein und beleuchtet die Vor- und Nachteile von Java. Darüber hinaus berichtet der vierte Abschnitt über eigene Erfahrungen mit der Ausbildung an den o.g. Institutionen. Die abschließende Zusammenfassung bewertet diese Erfahrungen.

2 Bisherige Erfahrungen mit Java in der Ausbildung

Bisherige Berichte über die Verwendung von Java in der Ausbildung stammen fast ausschließlich aus dem englischen und US-amerikanischen Raum. Die dortigen Erfahrungen sind sicherlich nur bedingt auf unsere Verhältnisse übertragbar, da dort eine andere Studiumsorganisation vorzufinden ist und sich daraus andere Anforderungen an die Lehre ergeben.

Lea, 1996 bildete den wahrscheinlich ersten Bericht über Lehrerfahrungen und wurde in Graci et al., 1997 erweitert. Die Autoren beschreiben sehr anschaulich ihre praktischen Erfahrungen mit dem Einsatz von Java in Grundstudiumskursen an der State University New York in Oswego und fassen zusammen, daß der Wechsel zu Java die Chance zur Neustrukturierung von Kursen bietet.

Battersby, 1997 berichtet über die Verwendung von Java in einem Kurs zu verteilter Programmierung an der Nottingham Trent University. Hervorgehoben werden gegenüber C++ die hohe Abstraktion von Details betriebssystemabhängiger Nutzung von Kommunikation und die elegante Einbindung von Nebenläufigkeit. Bemängelt wird das spezifische Strömekonzept zur Ein- und Ausgabe, sowie der Umfang der zu erlernenden Klassenbibliotheken. Die besonders hohe Motivation der Lernenden bei Java-Verwendung wird ebenfalls bemerkt.

Wallace et al., 1997 beschreibt die Einführung von Java als erste Programmiersprache in Lehre anstelle von Modula-2 an der University of West England in Bristol. Die Autoren erklären, daß Java gegenüber C++ und Smalltalk ohne

[1] Der Studiengang Informatik der TU Berlin bietet als Wahlpflichtveranstaltungen Praktika im dritten oder vierten Semester, Basislehrveranstaltungen im fünften Semester und Projekten in der zweiten Hälfte des Hauptstudiums and.

Frage die Sprache der Wahl ist. Sie heben hervor, daß Java ein hervorragendes Mittel zum Transport objektorientierter Konzepte der Programmierung ist. Problematisch schätzen sie die Lehrbuchsituation ein und berichten von zu hohen Erwartungen der Studierenden in Bezug auf Web-Applets, die in der Lehre nicht erfüllt werden können. Es erscheint ihnen perspektivisch von Vorteil, daß Java durchgehend auch für komplexere Themen wie nebenläufiger und verteilter Programmierung verwendet werden kann.

Garside, 1997 berichtet über sehr positive Erfahrungen mit Java in Kursen der University of Lancaster nach einem Wechsel von Ada. Die Autoren heben den guten Sprachentwurf hervor, die Integration objektorientierter Konzepte, die syntaktische Anlehnung an C, die erhöhte Motivation der Studierenden und das Potential zur Verwendung von Java über verschiedene Kurse auf unterschiedlichem Niveau. Ihre Erfahrungen bestätigen den Vorteil dieser Eigenschaften und lassen sich um die geringen Kosten für die technische Installation ergänzen.

Goedicke, 1997 stellt eine der wenigen Diskussionen der Verwendung von Java in der Grundausbildung von Studierenden im deutschsprachigen Raum dar (siehe auch Boehm et al., 1997). Der Bericht begründet sehr systematisch, warum Java im Vergleich zu Oberon, C++, Smalltalk und Beta für die Lehre ausgewählt wurde und bestätigt diese Auswahl durch positive Erfahrungen.

In King, 1997 findet sich eine Diskussion der von Sun hervorgegebenen Eigenschaften von Java vor dem Hintergrund der Informatikausbildung an der Georgia State University in Atlanta, USA. Die insgesamt sehr positive Beurteilung des Java-Einsatzes in der Lehre – erweitert um die Beobachtung geringer Kosten, hoher Motivation, der Übertragbarkeit auf Fortgeschrittenenkurse und der Nähe zum industriell höchst relevanten C++ – wird durch einige negative Punkte ergänzt. Vergleichsweise schlechte Ausführungszeiten, wenig Unterstützung durch Lehrbücher und die Dynamik der Sprachentwicklung werden angemerkt.

Schaller, 1997 faßt eine Podiumsdiskussion zu Java in der Lehre zusammen. Während die Teilnehmer verschiedene Probleme wie Umfang von Bibliotheken oder Schwächen der Ein- und Ausgabe benannten, überwogen die Vorteile des klaren Sprachentwurfs und die Beobachtung hoher Motivation der Studierenden.

Bowen, 1997 weist auf die strategische Bedeutung von Java in der Ausbildung für die Industrie hin. Während auf der einen Seite Sun durch die Verwendung in der Lehre auf eine Festigung der Marktstellung von Java hofft, scheint aber auf der anderen Seite auch zunehmend die Nachfrage der Industrie nach qualifizierten Java Spezialisten die Verwendung von Java verstärkt zu erfordern.

Die letzte Beobachtung läßt sich ohne weiteres auf die Situation im deutschen Raum übertragen. Nach Erfahrungen der Autoren wächst die Nachfrage nach Informatikern mit Java Kenntnissen enorm an und kann momentan vom Arbeitsmarkt nicht befriedigt werden.

Die meisten genannten Papiere berichten über Java in der Ausbildung im englischen und US-amerikanischen Raum. Aufgrund der anderen Struktur des dortigen Studiums und der erheblich stärkeren Standardisierung von Kursen lassen sich die Aussagen nicht einfach auf deutschsprachige Verhältnisse übertragen. Wir beschreiben im folgenden Abschnitt allgemeine Kriterien zur Auswahl einer

Programmiersprache für die Ausbildung, um anschließend über unsere Erfahrungen mit dem Einsatz von Java in der Ausbildung zu berichten.

3 Java für die Ausbildung

Objektorientierte Programmiersprachen existieren bereits seit etwa 30 Jahren, Simula seit 1968, Smalltalk seit 1972! Trotzdem sind sie noch nicht etablierter Bestandteil der Informatikausbildung. Dies liegt wahrscheinlich in erster Linie an der Schwerfälligkeit der Universitäten bzgl. der Lehrpläne bzw. Studien- und Prüfungsordnungen, aber mit Sicherheit nicht an der mangelnden Eignung. Erst in den letzten Jahren änderte sich dies aufgrund der Anforderungen aus der Industrie an Hochschulabsolventen.

Für die Ausbildung in objektorientierter Programmierung bieten sich neben einer Reihe forschungsorientierter Sprachen wie z.B. *Beta* momentan drei Programmiersprachen an. *Smalltalk* stellt die klassische objektorientierte Sprache dar, die sicherlich den konzeptuell saubersten Sprachentwurf hat. Smalltalk nutzt Objektorientierung in purester Form für die Modellierung primitiver Konstanten bis hin zu den abstrakten Höhen der Metaklassen. *C++* hat die höchste industrielle Relevanz für die Anwendungsprogrammierung, vor allem unter Windows. Für C++ existieren die meisten Entwicklungsumgebungen. *Java* schließlich ist in den letzten Jahren ein ernstzunehmender Kandidat für die Verwendung in der Ausbildung in objektorientierter Programmierung geworden.

Von diesen drei Programmiersprachen kommen für die Ausbildung nur Smalltalk oder Java in Frage. C++ ist u.a. wegen seiner Komplexität und der Vermischung imperativer und objektorientierter Konzepte für die Ausbildung nicht empfehlenswert. Die Autoren haben vor zwei Jahren begonnen, ihre Lehre von Smalltalk auf Java umzustellen. Für Java sprach eine Reihe von Gründen (siehe auch Boehm et al., 1997), wie statische Typsicherheit, industrielle Relevanz, Verfügbarkeit, Plattformunabhängigkeit und Motivation der Studierenden.

Darüber hinaus müssen die didaktische Eignung, der zu erwartende Lernerfolg, Perspektiven für weitere Lehre und spätere Berufstätigkeit sowie die technischen Möglichkeiten (Verfügbarkeit, Kosten, Installation) berücksichtigt werden.

Dies deutet an, wie komplex die Auswahl einer Programmiersprache für die Ausbildung ist. An der Entscheidung sind verschiedene Faktoren beteiligt. Dazu gehören die Lernenden, die Lehrenden, die Programmiersprache, die Beziehungen zwischen ihnen sowie der inhaltliche und infrastrukturelle Hintergrund. Dieser Abschnitt konzentriert sich auf Java als Lern- und Lehrmittel und die Beziehungen zwischen der Sprache und den Lernenden bzw. den Lehrenden. Lernende und Lehrende selbst und ihre Beziehungen werden hier nicht berücksichtigt.

3.1 Java als Lern- und Lehrmittel

Ziel der informatischen Ausbildung muß die Vermittlung von Konzepten und Fähigkeiten sein. Dementsprechend sind reine Programmierkurse lediglich als Fort- oder Weiterbildung anzusehen. Für die Vermittlung von Kenntnissen in

der objektorientierten Programmierung wird eine Sprache wie Java also zum Lern- und Lehrmittel und bildet nicht den eigentlichen Lerngegenstand.

Vorteile sind die statische Typsicherheit, der saubere Sprachentwurf, die syntaktische Anlehnung an C, woraus sich eine gute Grundlage für das spätere Erlernen von C++ ergibt. Weiterhin sind zu nennen die Einbindung von Nebenläufigkeit und Verteiltheit in die Sprache, die Abstraktion von betriebssystemnaher Programmierung und das Vorhandensein eines Dokumentationswerkzeugs und der Online-Dokumentation im JDK (s. Abschnitt 4.3). Schließlich sehen wir als Vorteil die akademische und kommerzielle Verbreitung bzw. industrielle Relevanz der Sprache und die weitgehende Unabhängigkeit von der technischen Infrastruktur und Plattformunabhängigkeit an.

Zu nennende Nachteile umfassen die Vermischung von imperativen und objektorientierten Anteilen bei den primitiven Datentypen und den Anweisungen und Unterschiede zwischen verschiedenen Plattformen, d.h. Probleme mit AWT, Zeilenendebehandlung und Dateinamen, die bedingt sind durch die Anbindung von Java an das lokale Betriebs-, Fenster- und Dateisystem. Weiterhin schien das spezifisches Strömekonzept für die Ein- und Ausgabe, als Nachteil, was aber durch den damit verbundenen Gewinn an Modularisierbarkeit aufgewogen wird. Schließlich sind zu nennen die schlechteren Ausführungszeiten im Vergleich zu nativem Code und die mangelhafte Lehrbuchsituation. Dagegen wirken sich die statische Typsicherheit, der saubere Sprachentwurf und die syntaktische Anlehnung an C insbesondere bei der Ausbildung von Anfängern positiv aus.

In der universitären Ausbildung stellt die objektorientierte Programmierung mit Java eine sinnvolle Ergänzung bzw. Vervollständigung des Grundstudiums dar, da in den meisten Fällen eine imperative Programmiersprache für die Ausbildung von Anfängern gewählt wird. Auf der kommerziellen Seite spricht die industrielle Relevanz dafür.

3.2 Java und die Lernenden

Wenn Java für die Ausbildung verwendet wird, werden zunächst blinkende Applets und weniger objektorientierte Programmierung oder Techniken erwartet. Die Tatsache, daß diese Erwartung in der Regel in der Anfängerausbildung nicht erfüllt werden kann und u.E. auch nicht erfüllt werden soll, ist der einzige Nachteil, den wir festgestellt haben.

Vorteile sind hingegen die Eigenschaften von Java als gute Grundlage für das Erlernen von C++, als gute Voraussetzung für eine spätere Berufstätigkeit, der geringe Installationsaufwand (gute Verfügbarkeit, kostenlos, einfache Installation) und die Unabhängigkeit von der technischen Infrastruktur durch Plattformunabhängigkeit.

Die Verfügbarkeit und Plattformunabhängigkeit ermöglichen die Arbeit der Lernenden am heimischen PC. Gleichzeitig machen sie relativ unabhängig von der technischen Infrastruktur.

Die Erwartungen der Lernenden können anfangs nur zum Teil erfüllt werden, da Anfänger Java in erster Linie mit animierten Applets und dem Internet bzw.

WWW in Verbindung bringen. Diese Themen gehören nach Meinung der Autoren nicht unbedingt in die Anfängerausbildung, wohingegen die aus der Sicht der Lernenden eher langweiligen theoretischen Konzepte der Objektorientierung unbedingt dazu gehören, um eine gemeinsame terminologische Basis zu schaffen. Dabei ist allerdings zu berücksichtigen, daß die Verwendung von Java für die Ausbildung kein Ersatz für objektorientierte Analyse, Modellierung oder Design sein kann. Diese Themen sind Gegenstand der Ausbildung von Fortgeschrittenen, für die Programmierkenntnisse eine sehr gute Voraussetzung sind.

Für Anfänger erleichtern Vorkenntnisse in C oder einer anderen imperativen Sprache den Einstieg (wünschenswert sind Datentypen, Variablendeklarationen und Anweisungen), sind allerdings nicht unbedingt erforderlich. Vorkenntnisse in C++ sind jedoch eher ein Nachteil für die Ausbildung mit Java. Zum späteren Erlernen von C++ ist Java jedoch als Voraussetzung sehr gut geeignet.

3.3 Java und die Lehrenden

Da Lehrende in gewisser Weise auch Lernende sind, was inbesondere aufgrund der dynamischen Entwicklung für die Ausbildung mit Java gilt, sei hier auch auf den vorhergehenden Abschnitt verwiesen.

Für die Lehrenden steht die didaktische Eignung einer Programmiersprache und ihre Abdeckung der Lehrinhalte im Vordergrund. Vorteile sind die Motivation der Lernenden, die Modularisierbarkeit in relativ überschaubare Inhaltspakete (s. Abschnitt 4.1) und die durchgängige Verwendbarkeit von Anfänger, über Fortgeschrittene bis hin zu Profis (von der Programmierausbildung bis hin zu Konzepten nebenläufiger und verteilter Programmierung, Komponentenkonzept oder Datenbankanbindung). Durch die letztgenannte Eigenschaft *skaliert* die Sprache mit der Erfahrung der Lernenden und der Komplexität des Lerninhalts (s. Abschnitt 4.2)

Nachteilig erscheinen die dynamische Sprachentwicklung (s. Abschnitt 4.3), deren Geschwindigkeit aber abnimmt, und ein Darstellungsproblem, das sich aus der teilweisen Komplexität und dem Umfang der Sprache ergibt (s. Abschnitt 4).

Die sehr hohe Motivation der Studierenden stellte die Autoren gleichzeitig vor das Problem, in den Veranstaltungen an der TU Berlin für Anfänger mit bis zu 230 Teilnehmern fertig werden zu müssen. Dies ist ein deutlicher Beleg für die Attraktivität von Java für die Lernenden.

Als Lehrmittel ist Java ebenfalls sehr attraktiv. Dies liegt im wesentlichen an seiner Modularisierbarkeit und Skalierbarkeit. Diese ermöglichen die Verwendung von Java für die Ausbildung von Anfänger, über Fortgeschrittene bis hin zu Profis.

4 Erfahrungen in der Ausbildung mit Java

Die im vorherigen Abschnitt diskutierten Argumente betreffen den Einsatz von Java für die Ausbildung allgemein. Ist diese Entscheidung gefallen, ergeben sich eine Reihe von Detailbeobachtungen, die wir in diesem Abschnitt darstellen.

4.1 Planung

Die Abbildungen 1 und 2 zeigen zwei mögliche Programme für jeweils dreitägige Kompaktkurse für Anfänger und Fortgeschrittene. Die von den Autoren als Praktikum durchgeführte Lehrveranstaltung an der TU Berlin folgt in etwa dem ersten Schema – durch die Organisation mit zwei Vorlesungs- und Übungsstunden pro Woche ergibt sich natürlich eine andere zeitliche Aufteilung.

Block	1. Tag	2. Tag	3. Tag
1	OO Konzepte, Imperative Konstrukte in Java	Klassen in `java.lang` und `java.util`	AWT, Ereignisse
2	Einführung Infrastruktur Übung: Erstes Programm mit `main` Methode	Übung: Online-Dokumentation, Daten konvertieren, Datenstrukturen nutzen und erweitern	Übung: Umgang mit AWT Komponenten
3	OO Konstrukte in Java	Ein-/Ausgabe mit `java.io` Klassen	Übung: Miniatur Anwendung mit GUI
4	Übung: Erste Java Klassen mit Vererbung	Übung: Eingaben ausgeben, Daten per Datei austauschen, Datenstruktur speichern/laden	Puffer, Ausblick, Verweise auf Online-Quellen

Abbildung 1. Ein Kompaktkurs zur Einführung in OOP mit Java

Block	1. Tag	2. Tag	3. Tag
1	AWT, Ereignisse	Netzwerkzugriff mit `java.net`	Java Beans
2	Übung: Grafikeditor, Spiel	Übung: Namensauflösung, Web-Client, Web-Server	Übung: Uhrwerk- und Anzeige-Beans
3	Applets	Verteiltheit mit RMI	JDBC
4	Übung: Grafikeditor, Spiel	Übung: Chat-Server, -Client, -Roboter	Übung: Zugriff auf mSQL und Access

Abbildung 2. Ein Kompaktkurs für Fortgeschrittene über drei Tage

Bei der Beispielplanung für einen dreitägigen Kompaktkurs wird die Hälfte der Zeit für Übungen am Rechner eingeplant. Nach der Erfahrung der Autoren ist dies die untere Grenze. Für das Erlernen des Sprachkerns ist je nach Vorkenntnissen unterschiedlich viel Übungszeit einzuplanen – das Erlernen und Einüben

der Syntax stellt eine ernste Hürde am Anfang dar. Der Umgang mit den Standardpaketen sollte – wie unten geschildert – in jedem Fall stark auf selbsttätiges Studium der Dokumentation und praktische Übungen gestützt werden.

In den Fällen, in denen ein Kompaktkurs am Ende nicht den gewünschten Lernerfolg hatte, war nach Erfahrung der Autoren zumeist die mangelnde Zeit für Übungen das Hauptproblem. Ein Verzicht auf weiterführende Vortragsblöcke zugunsten von Rechnerübungen war sinnvoll.

In den Kursteilen zur Darstellung der Standardklassen wurde schnell deutlich, daß lediglich ein Überblick möglich ist und die Grundkonzeptionen präsentiert werden können. Verantwortlich dafür ist einerseits der schiere Funktionsumfang der Standardpakete, deren detaillierte Behandlung jeglichen Zeitrahmen sprengt. Zum anderen kann es didaktisch nicht verantwortet werden, die Lernenden mit Auflistungen von Funktionsschnittstellen zu überfordern.

Es stellte sich heraus, daß die Vermittlung der Standardpakete auf den eigenständigen Umgang mit der Online-Dokumentation und auf die praktische Verwendung von Funktionen in Übungen gestützt werden sollte. Dazu muß vor der Darstellung der Bibliotheken eine Einführung in die Struktur und die Benutzung der JDK Dokumentation gegeben werden. Das Aufsuchen bestimmter Dokumentationsteile sollte zu einer der ersten Übungsaufgaben gemacht werden.

Neben der quantitativen Darstellungsgrenze trifft man an verschiedenen Stellen der Sprache auch auf qualitative Anforderungen, die eventuell jenseits des Kursniveaus liegen. So sind nebenläufige Threads zwar in hoher konzeptioneller Eleganz Bestandteil des Sprachkerns, werden aber kaum in einem Anfängerkurs hinreichend zu behandeln sein. Einen ähnlich klaren Niveaubruch finden man mittlerweile in Bezug auf die verteilte Programmierung mit RMI.

Diese Schwierigkeit wird schon beim ersten Beispielprogramm deutlich: Da man weder den AWT noch das `applet` Paket hinreichend einfach erklären kann, wird man sich in den Übungen sehr lange auf textbasierte Java Applikationen beschränken. Die allerersten Schritte zum Üben der imperativen Sprachanteile können praktisch nur durch Abläufe innerhalb der statischen `main` Methode gemacht werden, da noch keine eigenen Klassen definiert werden können.

Über Klassen hinaus bietet Java Pakete zur Strukturierung und Sichtbarkeitssteuerung in Klassensammlungen an. Die Autoren haben die Erfahrung gemacht, daß in Anfängerkursen dieses Konzept nicht eingeführt werden muß und eine einfache Erläuterung der Syntax des `import` Statements ausreicht.

4.2 Skalierbarkeit und Modularisierung

Aus den unterschiedlichen Niveaus der in Java eingebauten oder durch Klassenbibliotheken standardisierten Konzepte ergeben sich allerdings keineswegs nur Probleme, sondern auch einer der wichtigsten Gründe für den Einsatz von Java in der Ausbildung: Eine *Skalierbarkeit* der Sprache über Anforderungen hinweg.

Damit meinen wir, daß Java als erste Sprache für das Erlernen imperativer und objektorientierter Programmieren sehr gut geeignet ist, aber im Fortgang der Ausbildung als Implementierungsplattform beibehalten werden kann.

Die Sprache erweist sich als geeignet, um weiterführende Konzepte wie Nebenläufigkeit und Verteiltheit zu transportieren, bis hin zur Verwendung in komplexen Anwendungsprogrammen. Abbildung 3 zeigt diese Skalierbarkeit, die in Weisweber und Tolksdorf, 1998 im Detail diskutiert wird.

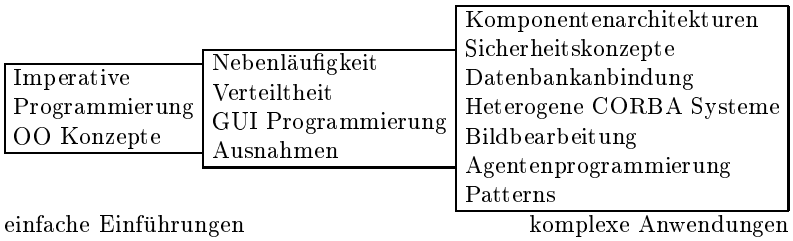

Abbildung 3. Die Spannbreite des möglichen Java-Einsatzes

Während dies im Hinblick auf eine Verwendung von Java durch das Studium hindurch große Chancen bietet, kann es dazu führen, daß Lerneinheiten Konzepte auf unterschiedlichen Niveaus vermischen. Es hat sich daher als vorteilhaft herausgestellt, zu Beginn lediglich textbasierte Java-Applikationen zu schreiben und komplexe Aspekte wie die umfangreiche AWT Bibliothek oder die Verwendung von Threads in Applets erst viel später einzuführen.

Ermöglicht wird dies erst durch eine weitere Eigenschaft der Sprache: *Modularisierbarkeit*. Darunter verstehen wir die Möglichkeit, Inhalte in klar abgegrenzten Teilen darzustellen und Java jeweils mit Erfolg dafür einzusetzen. Die Modularierbarkeit befördert eine klare Gliederung von Kursen, ermöglicht schnelle Lernerfolge sowie praktische Umsetzung und ist notwendige Voraussetzung für die genannte Skalierbarkeit.

4.3 Durchführung mit dem JDK

Praktische Übungen mit Java sind einerseits aufgrund der JDK Verfügbarkeit sehr einfach auf unterschiedlichsten Plattformen kostengünstig zu realisieren. Andererseits treten bei der Durchführung verschiedene technische Hürden – wie die Anpassung der Systemumgebung durch Pfadänderungen etc. – auf, die die Mängel dieser Umgebung für die Ausbildung deutlich machen.

Der JDK bietet eine für erfahrene Programmierer vollständige Entwicklungsumgebung mit Compiler und Debugger an. Für Anfänger fehlt eine integrierte Umgebung, die beispielsweise den Aufruf des Compilers per GUI-Button und ohne Umgang mit einer Kommandozeile ermöglicht. Schwierig zu vermitteln sind auch die Zusammenhänge, die manuelle Neucompilierung von Klassen erfordern. Eine Oberfläche mit einem integrierten und erweitertem `make` ist erforderlich – auch mit Blick auf die bei RMI-Verwendung noch komplexeren Abhängigkeiten.

Eine weitere Beobachtung betrifft die Verwendung von Editoren. Java Programme lassen sich mit jedem Texteditor erstellen. Allerdings sind Hilfsmittel

wie farbliche Hervorhebung von Schlüsselworten, Markierung von Klammernpaaren und – bei Anfängern besonders wichtig – automatische Einrückungen dann zumeist nicht vorhanden. In einer speziell vorbereiteten Umgebung wird sicherlich der `emacs` Editor mit entsprechendem Java-Modus die erste Wahl sein.

Dringend erforderlich für praktische Übungen ist die Ergänzung des JDK mit einer integrierten Entwicklungsumgebung. Sie ließe sich plattformübergreifend aus frei erhältlichen Tools konfigurieren oder auf Java Basis neu entwickeln.

Positiv hervorzuheben ist `javadoc`, das sehr gut zur Vermittlung von Aspekten der Code-Dokumentation einzusetzen ist und – bei entsprechender Einführung – durch die ansprechende Form der Dokumentation gute Lernerfolge verschafft.

Die Dynamik der Java Entwicklung führt zu technischen Problemen, falls mehrere JDK Versionen gleichzeitig installiert sind. Die Kurse der Autoren basierten sehr lange auf Java 1.0 und dem entsprechenden JDK, obwohl die Sprachversion 1.1 schon in den ersten *final* Versionen erhältlich war. Der so gewonnenen Stabilität im Kursprogramm stand eine Instabilität der technischen Umgebung gegenüber, die vom korrekten Setzen verschiedener Umgebungsvariablen und dem Auftreten von Pfaden in der richtigen Reihenfolge darin abhängig war.

Die mittlerweile auf JDK 1.1 umgestellten Kursinhalte, Beispiele und Übungen leiden besonders in den Anfängerkursen unter den Warnungen, die der JDK Compiler bei der Verwendung von Methoden der Standardbibliothek ausgibt, von deren Verwendung als *deprecated* abgeraten wird. Daß Methoden zwar vorhanden sind, ihre Verwendung aber unerwünscht ist, läßt sich Anfängern kaum vermitteln und trifft bei Beispielen auch öfter die Lehrenden recht unverhofft.

4.4 Lehrbücher

Lehre stützt sich immer auch auf Lehrbücher vom freien Markt. Im Hinblick auf die Informatikausbildung auf hohem Niveau ist die Situation dabei allerdings weiterhin enttäuschend. Nach wie vor bietet der Buchmarkt zwar sehr viele Titel zum Programmieren in Java, allerdings sind bisher nach Auffassung der Autoren noch keine didaktisch brauchbaren Bücher in deutscher Sprache erhältlich. Notwendig wären Lehrbücher, die Java als Mittel zum Erlernen von objektorientierter Programmierung darstellen und sich keinesfalls in der Beschreibung von Applet-Programmierung erschöpfen. Leider führen solche Titel auch zu falschen Erwartungshaltungen bei den Lernenden.

5 Schlußfolgerungen

In den vorhergehenden Abschnitten haben die Autoren eine Reihe von Beobachtungen zum Einsatz von Java in der Ausbildung zusammengetragen und über Erfahrungen berichtet. Dabei wogen weniger die informatischen Eigenschaften der Sprache positiv, als Umstände ihrer Verwendung in der Ausbildung.

In der Tat lassen sich wenig Gründe für einen Wechsel beispielsweise von Smalltalk nach Java aus den reinen Spracheigenschaften ableiten – wahrschein-

lich ist Smalltalk in der Konzeption klarer. Der technische und perspektivische Kontext von Java liefert hingegen viele Argumente dafür.

Zwei zentrale Vorteile von Java sind die Modularisierbarkeit und die Skalierbarkeit. Modularisierung beinhaltet hier zwei Aspekte. Einerseits die Modularisierbarkeit von Software (für die Lernenden) und anderseits die der Inhalte (für die Lehrenden). Sie ermöglichen Teamarbeit bzw. eine flexible Gestaltung von Kursen nach dem Baukastenprinzip. In Verbindung mit der Skalierbarkeit erlaubt Java ein durchgängiges konsistentes Konzept für die Ausbildung vom Anfänger bis zum Profi. Weiterhin ist nicht nur Java selbst eine gute Voraussetzung für eine spätere Berufstätigkeit, sondern ist ebenfalls gute Grundlage für das Erlernen von C++, das zur Zeit die höchste industrielle Relevanz aufweist.

Ein großes Problem bei der Ausbildung war bis jetzt die dynamische Entwicklung von Java. Die Geschwindigkeit der Veränderungen wird sich aber in Zukunft verlangsamen. Mit Schwierigkeiten verbunden waren auch die unterschiedlichen Vorkenntnisse der Teilnehmer. Durch geeignete Wahl der thematischen Bausteine läßt sich dies zumindest abmildern. Zu diesen Bausteinen gehören die Konzepte der Objektorientierung sowie die Aufteilung bei der Darstellung von Java in imperative und objektorientierte Anteile. Letzteres hat einerseits den Vorteil, daß je nach den Vorkenntnissen der Teilnehmer die imperativen Anteile wegfallen können. Anderseits führt es zu dem konzeptionellen und didaktischen Vorteil einer sauberen Trennung beider Programmierparadigmen.

Die Verwendung von Java kann zu zu hohen Erwartungen bei den Lernenden führen. Aus diesem Grund sollte man am besten schon in der Ankündigung klar sagen, daß Java nicht der Lerngegenstand ist, sondern Mittel zur Vermittlung objektorientierter Programmiertechniken und -methoden. Es sollte deutlich werden, daß graphische Benutzerschnittstellen, Applets und alle anderen weiterführenden Konzepte am Anfang nicht das Wichtigste sind. Das genannte Zeitproblem bei der Durchführung von Kursen erzwingt teilweise einen Verzicht auf die Darstellung dieser Themen.

Die Autoren haben bei der Ausbildung mit Java am meisten ein gutes Lehrbuch und eine in Java geschriebene und kostenlose integrierte Entwicklungsumgebung vermißt. Letzterer Mangel wirkt sich momentan sehr negativ auf die Durchführung von Kursen aus. Wenn die Entscheidung für Java in der Ausbildung einmal gefallen ist, empfehlen die Autoren die Beachtung folgender Punkte:

- Die technische Infrastruktur (Betriebssystem, Editor, ...) sollte den Teilnehmern vor Beginn der Ausbildung bekannt sein. Während der Ausbildung bleibt dazu in der Regel keine Zeit. Eine Ausbildung mit „blutigen“ Informatikanfängern macht also wenig Sinn.
- In der Ausbildung sollte zu Beginn eine Einführung in die grundlegenden Konzepte der Objektorientierung gemacht werden (Objekte, Klassen, -hierarchien, Vererbung, Metaklassen, Kapselung, Polymorphie, Bindung etc.).
- Die Darstellung von Java sollte in imperative und objektorientierte Anteile aufgeteilt werden. Bei entsprechenden Vorkenntnissen der Teilnehmer können die imperativen Anteile wegfallen.

- Eine Anfängerausbildung sollte sich neben der Objektorientierung in erster Linie mit allgemeinen Programmierkonzepten befassen. Für Java gehören dazu die primitiven Datentypen, Anweisungen und Deklarationen sowie die Pakete `lang`, `io` und `util`. Alle anderen Pakete sollten Bestandteil weiterführender Ausbildungsabschnitte sein.
- An den Anfang der Ausbildung gehört auch eine Einweisung in den Umgang mit der Klassendokumentation.
- Eine Ausbildung sollte etwa zur Hälfte aus der Vermittlung von Inhalten und zur anderen Hälfte aus praktischer Arbeit bestehen.

Trotz einiger kritischer Anmerkungen hat die Verwendung von Java in unseren Ausbildungsveranstaltungen einen sehr positiven Eindruck sowohl bei den Lernenden als auch bei den Lehrenden hinterlassen. Aufgrund seiner Modularisierbarkeit und Skalierbarkeit ermöglicht Java ein durchgängiges inhaltliches und didaktisches Konzept, so daß die Sprache in sämtlichen Ausbildungsabschnitten eingesetzt werden kann. Die Vielzahl der Perspektiven der Spache Java und der Objektorientierung für die berufliche Praxis hat die Autoren weiterhin von der Richtigkeit der Wahl von Java für die Ausbildung bestärkt.

Literatur

Battersby, A. (1997). Initial Experience with Java in a Distributed Computing Module. *Monitor*, (8).

Boehm, M., Freytag, J., Owsnicki-Klewe, B., Pfeiffer, G., und Raasch, J. (1997). Objektorientierung in der Informatikausbildung auf der Basis von Smalltalk. *Informatik-Spektrum*, 20(6):335–343.

Bowen, B. D. (1997). Educators embrace Java. *Javaworld*, (1):Online Dokument. *http://www.javaworld.com/javaworld/jw-01-1997/jw-01-education.html*.

Garside, R. (1997). Teaching Java at Lancaster. *Monitor*, (8).

Goedicke, M. (1997). Java in der Programmierausbildung: Konzept und erste Erfahrungen. *Informatik-Spektrum*, 20(6):357–363.

Graci, C., Lea, D., und Mohammadi, R. (1997). Experiences using Java at SUNY Oswego. *Dr Dobb's Journal*, (Fall special issue).

King, K. (1997). The Case for Java as a First Language. In *Proceedings of the 35th Annual ACM Southeast Conference*.

Lea, D. (1996). Some Questions and Answers about using Java in Computer Science Curricula. Online Dokument. *http://g.oswego.edu/dl/html/javaInCS.html*.

Schaller, N. C. (1997). Using Java in Computer Science Education. In *SIGCSE/SIGCUE Conference on Integrating Technology into Computer Science Education*, pages 140–142, Uppsala, Sweden. *http://www.cs.rit.edu/~ncs/Uppsala97/index.html*.

Wallace, C., Martin, P., und Lang, B. (1997). Not whether Java but how Java. *Monitor*, (8).

Weisweber, W. und Tolksdorf, R. (1998). Durchgängiger Einsatz von Java in der Informatikausbildung: Von Einführungen bis zu offenen verteilten Anwendungen. In *Proceedings Smalltalk und Java in der Ausbildung, STJA98*.