Using Java in the Undergraduate Computer Science Curriculum

Konstantin Läufer

Department of Mathematical and Computer Sciences
Loyola University Chicago
6525 North Sheridan Road
Chicago, IL 60626, USA
laufer@math.luc.edu
http://www.math.luc.edu/~laufer/

February 24, 1997

Abstract

Since its initial release two years ago, the programming language Java has gained tremendous popularity and momentum. Java was introduced by Sun within the context of the recent growth of the World Wide Web chiefly as a technology for providing interactive web content. However, Java is also a general-purpose object-oriented programming language in the spectrum between two other popular object-oriented languages, C++ and Smalltalk. This paper discusses past experiences and future plans for integrating Java into the undergraduate computer science curriculum.

Keywords: curriculum issues; CS1/2; object-oriented technology; programming languages for the first course

1 The Design of Java

A major design objective for the Java language [GJS96] was to combine a complete but minimal set of existing features. Java is an object-oriented, concurrent, distributed, platform-independent programming language. This combination of paradigms is crucial for supporting interactive web content. The object-oriented approach facilitates structuring programs through composition and inheritance. Concurrency is generally needed in interactive applications to allow an activity to proceed while responding to user input. Distributed programming provides the capability to exchange information across a network. Platform independence allows the same application to run across networks such as the Internet that are heterogeneous with respect to hardware and operating systems.

Java was specifically designed to be a safe and simple language. Obscure or dangerous language features, such as pointer arithmetics and arbitrary pointer casts [ES90], have purposely been left out. Static typing enables the compiler to guarantee that certain programs will not suffer from type-related run-time errors, such as a method not understood by a receiver. Garbage collection relieves the programmer from having to micromanage dynamically allocated memory; the system automatically detects when data is no longer used and recycles it. Memory leaks, a very common form of programming error, no longer occur.

2 Java in the Undergraduate Curriculum

Java can be employed in a variety of places in the curriculum, from the introductory computer science sequence (CS1/2) to an intermediate course in object-oriented programming to specialized courses such as networking, software engineering, and graphical user interfaces. All courses potentially benefit from focusing on concepts rather than details by keeping the general level of abstraction high. This section presents a (non-inclusive) list of undergraduate courses with small Java examples for some of the courses.

CS1/2

Those who have shifted the introductory computer science courses from Pascal [Cat96] to C++ [ES90] during the past couple of years are likely to appreciate the richer set of program structuring mechanisms in C++. Nevertheless, some of us, including the author, miss the simplicity of Pascal. In Java [GJS96], garbage collection together with an ample collection of predefined types such as strings and smart arrays considerably reduce the level of syntactic detail that has to be mastered before meaningful programs can be written. This allows the instructor to spend more time on concepts.

The first example shows an iterative and a recursive version of a factorial function. Static functions roughly correspond to ordinary functions in other languages.

```
class Factorial {
    public static void main() {
        System.out.println(iterFac(5));
        System.out.println(recFac(5));
    }

    static int iterFac(int n) {
        int result = 1;
        while (n > 0) {
            result *= n;
            n --;
        }
        return result;
    }

    static int recFac(int n) {
        if (n <= 1) {
            return 1;
        } else {
            return n * recFac(n - 1);
        }
     }
}
```

The next example shows a linked-list based stack class with an auxiliary node class. Nodes removed from the stack are garbage-collected eventually by system. Instance variables such as `topNode` are initialized to the `null` object. All identifiers of class type are references (implicit pointers).

```
class Node {
    public Node(Object item) {
        this.item = item;
        this.next = null;
    }

    public Node(Object item, Node next) {
```

```
            this.item = item;
            this.next = next;
        }

        public Object item;
        public Node next;
    }

    class Stack {
        public void push(Object item) {
            topNode = new Node(item, topNode);
        }

        public void pop() {
            topNode = topNode.next;
        }

        public Object top() {
            return topNode.item;
        }

        public boolean empty() {
            return topNode == null;
        }

        Node topNode;
    }
```

These two classes are for illustrative purposes only, since the `java.util` package already contains a `Stack` class and a more general `Vector` class. *Packages* are groups of related classes.

Object-oriented programming

Since Java is syntactically very close to C++, the transition from CS1/2 in C++ to Java at the intermediate undergraduate level is also facilitated. An undergraduate course in object-oriented programming in the second year would offer an adequate context for introducing Java.

Unlike C++ [ES90], Java [GJS96] provides a separate *interface* construct for the specification of abstract data types. This allows some amount of separation between interface and implementation hierarchies. Every class has exactly one superclass (singe inheritance) and zero or more interfaces. Since interfaces do not provide any code, this approach avoids the complexity of multiple inheritance in C++.

Networking

Advanced, specialized undergraduate courses can similarly benefit from using Java. In a networking course, various predefined socket and URL classes facilitate writing network clients and servers. The feasibility of using Java for networking has been established through various projects such as the Java Server [Sun96c]. Java supports concurrency and provides facilities for communication and synchronization of threads. Java also supports distributed programming through Remote Method Invocation [Sun96b], which includes an interface definition language (IDL) for Java.

Compiler construction

A compiler course could benefit from using Java in various ways. Front-end tools for the genera-

tion of lexical analyzers and parsers [Sun96] are available. Furthermore, the Java virtual machine [LY97] provides a platform-independent target architecture for compilation.

Software engineering

In a software engineering course based on an object-oriented analysis and design method such as Object Modeling Technique [Rum91], the class relationships established during analysis and design easily translate to a Java implementation. Software design patterns [GHJV95] can be expressed conveniently in Java as well. In addition to the standard Java class libraries [GYT96], powerful libraries of reusable components such as the Java Generic Library [Obj97] are available as well.

User interfaces

In a course on graphical user interfaces, the Java abstract window toolkit (AWT) provides an easy-to-use, platform-independent abstraction of event-driven graphical user interfaces. The following example [Fla96] shows a simple applet for scribbling with the mouse. By using the existing packages for drawing and handling mouse events, the new code is very small.

```
import java.applet.*;
import java.awt.*;

public class Scribble extends Applet {
    private int last_x = 0;
    private int last_y = 0;

    // called when the user clicks
    public boolean mouseDown(Event e, int x, int y)
    {
        last_x = x; last_y = y;
        return true;
    }

    // called when the mouse moves with the button down
    public boolean mouseDrag(Event e, int x, int y)
    {
        Graphics g = getGraphics();
        g.drawLine(last_x, last_y, x, y);
        last_x = x;
        last_y = y;
        return true;
    }
}
```

3 Teaching Resources for Java

An important issue remains whether adequate teaching resources for Java are available, considering that the language is rather new [Lea96]. For the past two years, the rate at which books in Java have been published has grown spectacularly. While many of these books are not appropriate as college-level textbooks, several publishers now have a Java series and offer several texts for CS1/2 as well [Dei97]. Integrated development environments (IDEs) with user interface builders are now available from the usual contenders as well as from Sun itself; many Java IDEs are based on existing ones for C++. Java programs compile and run on virtually all major platforms, including most Unix versions, Windows 95/NT, MacOS, and, as of recently, Windows 3.1. Command-line compilers and interpreters for these platforms are available for free [Sun96a]. Additional

resources such as code repositories, tutorials, and newsgroups are available on the Internet.

4 Experience in Teaching with Java

The author has taught a variety of computer science courses since 1991. In 1994, the author developed and first held an intermediate course in object-oriented programming using C++, while CS1/2 were shifted from Pascal to C++. The author plans to offer a similar course in Java in summer 1997 and proposes shifting CS1/2 to Java by 1997.

At the graduate level, the author has used Java with excellent results in an object-oriented programming course [Läu97a] involving three projects. Student productivity has increased considerably by switching away from C++. In a graduate course in software engineering [Läu97b], the author is currently offering students a choice between C++ and Java as the implementation language of the semester-long group project.

The author has also had positive experiences in teaching short courses in Java for professionals.

References

[Cat96] Bill Catambay. *Pascal Standards*. August 1996. Available from http://www.catambay.com/pascal-central/standards.html.

[Dei97] H. M. Deitel and P. J. Deitel. *Java – How to Program*. Prentice-Hall, 1997.

[ES90] M. Ellis and B. Stroustrup. *The Annotated C++ Reference Manual*. Addison-Wesley, 1990.

[Fla96] David Flanagan. *Java in a Nutshell*. O'Reilly, 1996.

[GHJV95] Erich Gamma, Richard Helm, Ralph E. Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley, Reading, Massachusetts, 1995.

[GJS96] James Gosling, Bill Joy, and Guy Steele. *The Java Language Specification*. Sun Microsystems, Mountain View, California, Version 1.0 edition, August 1996. Available from http://java.sun.com/doc/language_specification/.

[GYT96] James Gosling, Frank Yellin, and The Java Team. *Java API Documentation*. Sun Microsystems, Mountain View, California, Version 1.0.2 edition, April 1996. Available from http://java.sun.com/products/JDK/1.0.2/api/.

[Läu97a] Konstantin Läufer. *Comp 473: Object-Oriented Programming*. January 1997. Available from http://www.math.luc.edu/~laufer/courses/473/.

[Läu97b] Konstantin Läufer. *Comp 474: Software Engineering*. January 1997. Available from http://www.math.luc.edu/~laufer/courses/474/.

[Lea96] Doug Leah. *Some Questions and Answers about using Java in Computer Science Curricula*. November 1996. Available from http://gee.cs.oswego.edu/dl/html/javaInCS.html.

[LY97] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification*. Addison-Wesley, 1997.

[Obj97] Java Generic Library 2.0. January 1997. Available from http://www.objectspace.com/.

[Rum91] James Rumbaugh et al. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

[Sun96] The Java Compiler Compiler. February 1997. Available from http://

www.suntest.com/Jack/.

[Sun96a] Java Developers Kit Release 1.0.2, May 1996. Available from http://java.sun.com/products/JDK/.

[Sun96b] Java Remote Method Invocation Prebeta, November 1996. Available from http://chatsubo.javasoft.com/current/rmi/.

[Sun96c] Java Server. May 1996. Available from http://jeeves.javasoft.com/products/java-server/.