IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

#ITP4365

A PERSONALIZED SYSTEM OF INSTRUCTION FOR TEACHING JAVA

Henry H. Emurian
Department of Information Systems, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, Maryland 21250 USA
Telephone: 410-455-3655; Fax: 410-455-1073; E-mail: emurian@umbc.edu

Ashley G. Durham[1]
Health Care Financing Administration, 7500 Security Boulevard, mailstop N2-13-16, Baltimore, Maryland 21244 USA
Telephone: 410-786-5775; Fax: 410-786-0271; E-mail: adurham@hcfa.gov

INTRODUCTION

This paper addresses the challenge of how to structure a learning environment to teach object-oriented computer programming to students who may need an introductory course in that discipline but who may lack the experiences to use symbol manipulations with confidence. In contrast to computer science students, information systems students sometimes exhibit these latter attributes, but they would nonetheless benefit professionally from acquiring rudimentary programming language knowledge and skill. To accomplish that objective, the Personalized System of Instruction (PSI), originally developed by Keller (1968), is described here to foster equivalent competence among students in an initial Java‰ coding assignment in an introductory programming course. The intent of integrating a Java tutoring system into the PSI framework as the first laboratory exercise is to ensure that all students in the class have at least this background experience in common prior to the introduction of advanced features of interface implementation that are taught during the remainder of the semester. Self-report and performance data are presented to support the use of this pedagogical approach in the classroom.

The PSI methodology is based on the following five factors: (1) ***unit perfection***, in which progress from one step in learning to another step requires perfect performance in the prior step; (2) ***self-paced progression***, in which the student may move through a training experience at a self-determined rate; (3) ***focus on the written word***, in contrast to traditional lectures, to transmit information to the student; (4) ***repeated testing*** of concepts, and (5) ***collaborations and discussions*** with peers and experts. These five factors together constitute the PSI proposed by Keller (1968) and implemented by Ferster and Perrott (1968). Many studies support the effectiveness of the PSI (e.g., Kritch & Bostow, 1998), which contains features that are intended to meet the needs of the individual learner in ways that have long been known to overcome individual differences and to promote high achievement levels in all students (Bloom, 1984).

This paper reports the outcome of the use of the PSI in a graduate-level course that contains instruction in implementing graphical user interfaces with the Java Abstract Windowing Toolkit (AWT). The data reported here will show the use of the PSI to teach a class of students to write a Java Applet. The study extends our previous work, which validated a web-based tutoring system for training in fundamental aspects of Java by documenting improvements in programming confidence and competence immediately after students used the tutoring system (Emurian, Hu, Wang, & Durham, 2000). The present study broadens the number of assessment occasions to include a third assessment that occurred after the fifth PSI factor listed above had been completed by all students in a classroom discussion and collaboration setting. Finally, to assess the durability of learning, the study includes a fourth and final assessment that was administered during the last class of the semester, which occurred over three months after the third assessment occasion.

Our pedagogical approach emphasizes a programmed instruction methodology for implementing the first four factors in the PSI by means of the web-based tutoring system for Java training. Programmed instruction technology for teaching offers specific guidelines to follow in the construction of procedures that manage the moment-by-moment progress of a student during study events that are structured within the framework of a behavioral theory of learning (Skinner, 1958). The theoretical assumption is that the steps involved in learning a complex task, such as constructing a computer program, can be specified with sufficient precision that reinforcement contingencies can be applied to the component units that lead to task mastery. These ideas and concepts are grounded within the experimental analysis of behavior literature (e.g., Holland, 1960). This principle-based learning technology predates computer-based instructional systems, which now encompass a broad and multidisciplinary field of investigations and applications (Brock, 1997).

In the present tutoring system, the operational definitions of the information units to be learned by a student are based upon research in verbal learning that identifies at least three types of verbal information paradigms: (1) **item information**, which records the occurrence of events and is commonly tested by recognition tests; (2) **associative information**, which records relationships between separate events and is commonly tested by paired-associates tests; and (3) **serial order information**, which records the temporal sequence of a string of events and is commonly tested by serial recall tests (Li & Lewandowsky, 1995). The programmed progression among unit sizes studied here is based upon a functional account of verbal behavior, which suggests a systematic transition in learning from textual items to streams as a function of practice (Greer & McDonough, 1999). The adoption of the feature that learners be required to construct accurate responses by recalling units of increasing complexity is based upon earlier work by the authors in which learning and retention of UNIX‘ command sequences were superior under conditions of recall in comparison to recognition of textual information items (Durham & Emurian, 1998).

TUTORING SYSTEM DESIGN

The technical and operational details of the tutoring system have been presented elsewhere (Emurian, et al., 2000). The system consists of a series of Java Applets embedded within the WebCT‘ course management software that allows users to create guest accounts to access the system (http://webct.umbc.edu/public/JavaTutor/index.html). The design of the tutoring system is a synthesis of principles of programmed instruction (Skinner, 1958), verbal learning and memory (Anderson, 1995), the elaboration

This paper appears in the book, *Managing Information Technology in a Global Economy*, the proceedings of the Inforrmation Resour es Management Association International Conference. Edited by Mehdi Khosrow-Pour.

theory of instruction (Reigeluth & Darwazeh, 1982), practice and retention (Durham & Emurian, 1998), and instructional design (Tennyson & Schott, 1997).

The objective of the programmed instruction tutoring system is to teach a learner to construct a stream of 36 elements that together constitute a Java computer program that displays a text string in a Netscape" browser window. The approach taken is first to specify the terminal performance, which is to write the program correctly, and then to craft a series of programmed instruction steps that progress to that goal. The outcome is tested mastery of the meaning of each of the 36 elements and the interrelationships among those elements in the production of the terminal performance. The 36 elements are displayed in Figures 3 and 4, which are discussed below.

Overview

The learner progresses through the system in five stages. The full instruction set presented to the learner is available for observation within the online tutor.

Figure 1 shows the interfaces that the learner encounters during the first two stages of training. The left interface requires matching a displayed item. The right interface requires finding a displayed item in the list. These interfaces promote familiarity with the formal features of the symbols and with experience in detecting similarities and differences among the symbols.

Figure 1: The interfaces presented during the first two stages.

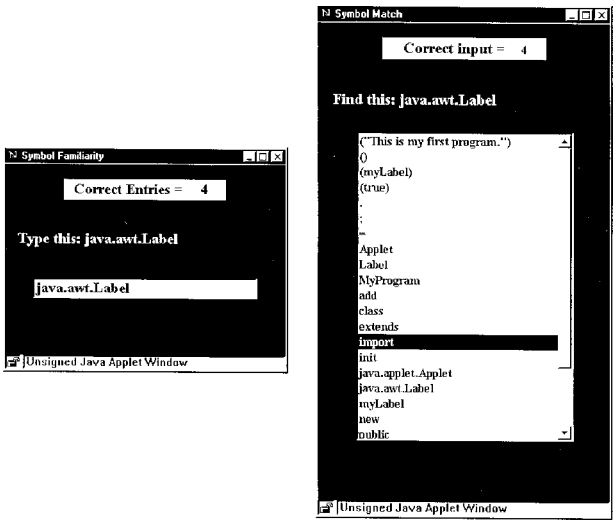
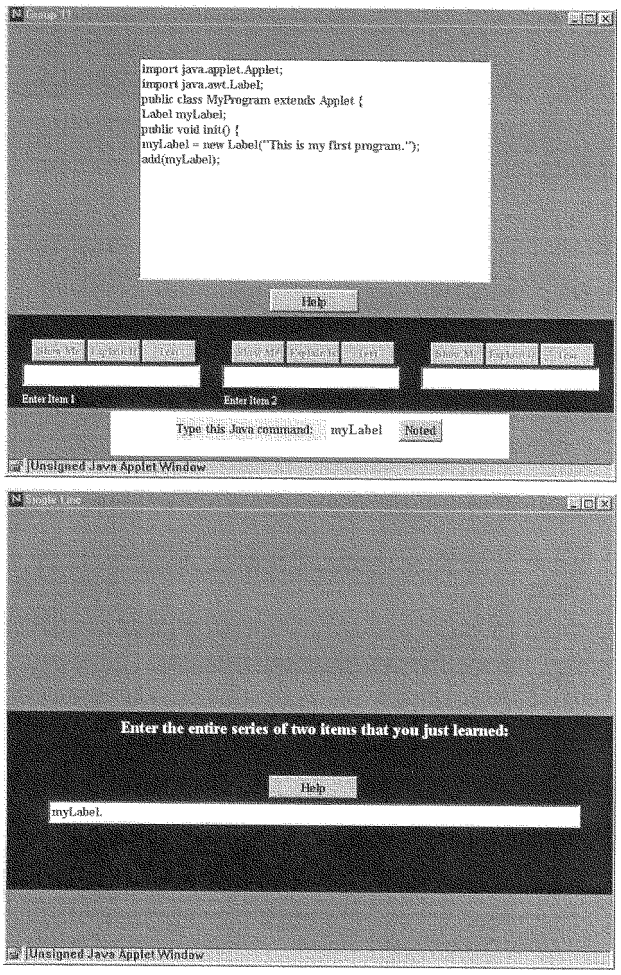


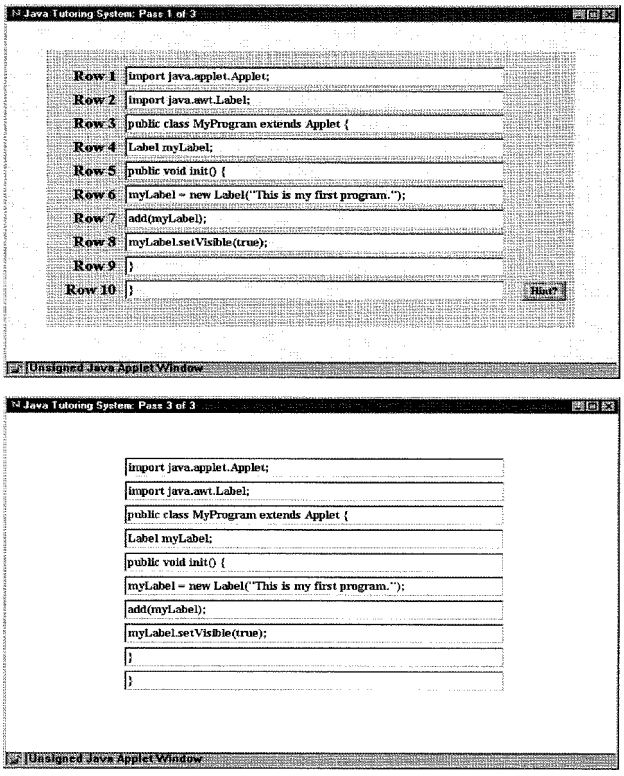
Figure 2 shows the two interface types that are presented during the third stage of the tutor. The top interface teaches the individual items of code, which are displayed cumulatively in the white space as the learner enters the items correctly. This interface teaches up to three items of code, entered into keyin fields at the bottom, and there are 14 of these interfaces in the tutor. This interface also requires passing a multiple-choice objective test on each item. Each successive item must be typed correctly in the keyin field from memory before the learner may progress to a subsequent item. Next, the bottom serial stream interface is presented, requiring entering the items that were just learned. If the input is incorrect, the learner recycles again through the item interface, and that cycle repeats until the code on a row is entered correctly. Then the next item interface is presented.

Figure 2: The two interface types that are presented during the third stage.



After the completion of the item and serial stream interfaces, a row by row interface is presented for the fourth stage. **Figure 3** shows two of the three row by row interfaces that are presented. The top view shows the first pass interface, and the bottom view shows the third pass interface. The second pass interface has small row labels on a white background. Each interface requires entering the lines of code correctly, row by row. The input on a row is based on the format of the program that was displayed in the white space in the item interface. The input on a row must be accurate before the next keyin field is enabled. If the user can not enter the row correctly, the item interface for the code in that row is presented again, and that cycle repeats until the code on a row is entered correctly.

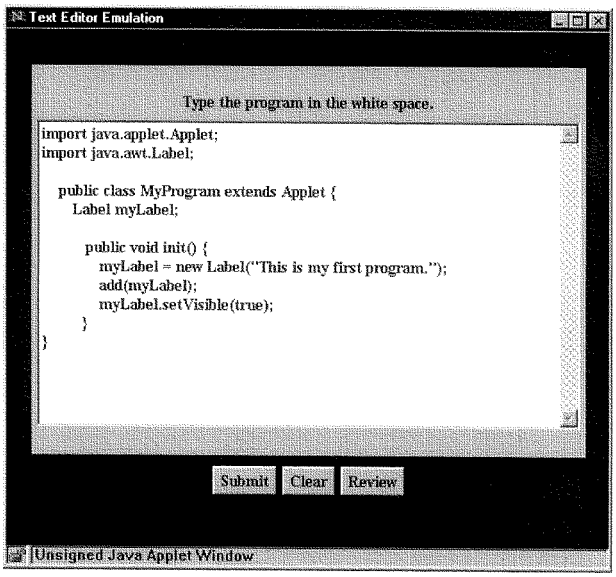
Figure 3: Two of the three row by row interfaces that are presented during the fourth stage.



During the first pass, the user may select an optional "Hint" window, which displays a description of the objective of the code in that row. Additionally, a multiple-choice test must be passed on the objective of each row. The second pass interface has the "Hint" window available, but the third pass interface does not have this option. Accordingly, the learner must complete three passes of entering the code in a row by row format. A justification of this number of passes is presented elsewhere (Durham & Emurian, 1998).

After the learner completes the row by row interfaces, a text editor emulation interface is presented for the fifth stage. **Figure 4** shows this interface with correct code entered. This interface relaxes the format that was enforced in the preceding interfaces, and it evaluates the input as a stream. If the learner is not able to enter the code correctly, the "Review" button initiates a recycling through the third pass of the row by row interface. Accordingly, the learner may recycle back to the row and item interfaces as required to master the code. This text editor emulation interface exits when the code is entered correctly. Although the tutor does present further instructions about the running of the Applet on the world-wide web (www), these details are presented in a lecture and discussion format for the classroom work reported here.

Figure 4: The text editor emulation interface presented during the fifth stage.



PROCEDURE

The tutor was presented as the first exercise in a graduate course (Fall, 2000) entitled "Graphical User Interface Systems Using Java." There were 12 graduate students in the class (eight females, median age = 27.5; four males, median age = 30). Prior to using the tutor, each student completed a questionnaire in which was presented two rating scales. The first rating scale assessed the student's prior experience with Java, and it consisted of the following instruction and five choices:

How would you describe your experience with Java as of this moment?

1 = No experience. (I am a novice in Java.)
2 = Some experience.
3 = Moderate experience.
4 = Much experience.
5 = Extensive experience. (I am an expert in Java.)

The second rating scale assessed the confidence that the student currently had in being able to use each of the 24 unique Java items to write a Java computer program. The rating scale consisted of the following instruction and five choices for each of the 24 items:

How confident are you that you can now use the following symbol to construct a Java program?

1 = Not at all confident. I do not know how to use the symbol.
2 = Only a little confident.
3 = Fairly confident.
4 = Very confident
5 = Totally confident. I know how to use the symbol.

The student was also asked to write a Java Applet to display a text string, as a Label object, in a browser window. Additionally, the questionnaire solicited demographic information. All data were collected and saved using the online assessment and recording features of WebCT‘.

At the conclusion of the two hours that were allotted to the tutoring system or whenever a student finished the tutor prior to that time, a post-tutor questionnaire was completed. The confi-

dence assessment scale and the writing of the Applet were repeated, and three additional rating scales were administered. The first scale instruction and choices were as follows:

What was your overall reaction to the tutor?

1 = Totally negative. I did not like the tutor.
2 = Only a little negative.
3 = Neutral.
4 = Only a little positive.
5 = Totally positive. I liked the tutor.

The second scale instruction and choices were as follows:

In terms of learning Java, how would you rate your experience in using the tutor?

1 = Totally negative. The tutor did not help me to learn Java.
2 = Only a little negative.
3 = Neutral.
4 = Only a little positive.
5 = Totally positive. The tutor did help me to learn Java.

The third scale instruction and choices were as follows:

How would you rate the usability of the tutor?

1 = Totally negative. The tutor was difficult to use.
2 = Only a little negative.
3 = Neutral.
4 = Only a little positive.
5 = Totally positive. The tutor was easy to use.

The students were then dismissed from the class, and the tutor continued to be available for those students who were motivated to access the tutor outside of class.

During the immediately succeeding class period, which occurred one week later, the instructor discussed the Applet code with the students using a lecture format ("chalk and talk"). The approach was to have the students enter the code into a text editor at the time the items were presented and discussed on the board. The www directory tree and HTML file were also presented and discussed. The students then compiled the Java code and ran the Applet in a Netscape Communicator browser by accessing the HTML file as a URL on the web. During these latter events, the students were encouraged to help each other and to seek help from the instructor and course assistant as needed. After all students ran the Applet on the web, they again completed the confidence ratings and the writing of the Applet code in the assessment questionnaire. This identical assessment was repeated during the fourteenth class, the final class of the semester.

RESULTS

At the conclusion of the class time allotted for completing the tutor, six students had finished all parts of the tutor, and six students were still working on the row by row or text emulation interfaces. A comparison of the magnitude of the changes between pre-tutor ratings and the next two rating occasions did not support differences between these two groups in changes for post-tutor ratings, $F(1,10) = 0.75$, $p > .10$, and post-applet ratings, $F(1,10) = 2.48$, $p > .10$. Accordingly, the two groups were pooled for the following analyses.

Figure 5 presents box-plots displaying median confidence ratings for all subjects for the 24 distinct items that were used to compose the program. A box plot is presented across the four assessment occasions: (1) **pre-tutor**, (2) **post-tutor**, (3) **post-applet**, and (4) **the final class**. The figure graphically shows the progressive increase in self-reports of confidence across the four assessment occasions. A MANOVA approach to comparing means based on the differences (*D*) between pairs of observations across the six combinations of assessment occasions for all subjects showed a significant effect of occasions, $F(5,66) = 18.12$, $p < .001$. Pairwise contrasts supported the conclusion that the most dependable changes in median confidence occurred over the first two assessment occasions. Since ordinal data are problematic for detecting and interpreting small effect size differences, replication is the preferred strategy for demonstrating the dependability of these observations.

Figure 5: Box plots showing median confidence ratings for all subjects for the 24 distinct items that were used to compose the program.PRE = Pre-Tutor, POST = Post-Tutor, APPLET = Post-Applet, and FINAL = Final Class.

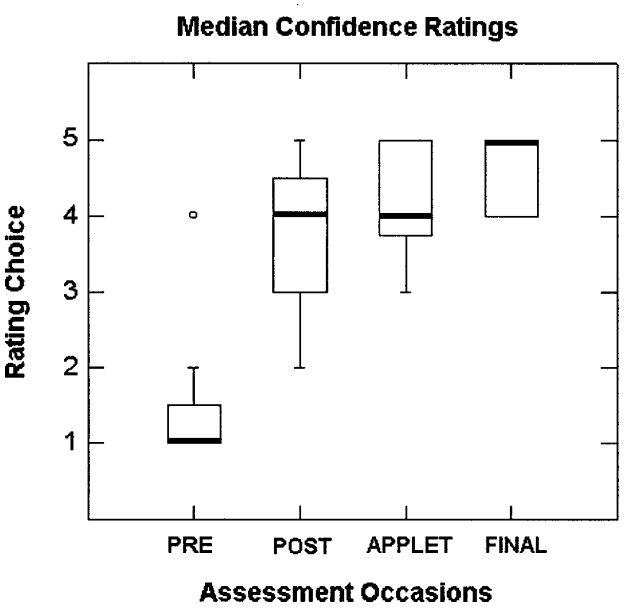


Figure 6 presents the total number of correct Java programs that were written into the questionnaire across the last three assessment occasions. The data are presented as a cumulative total. Since no subject wrote a correct program during the pre-tutor assessment, the figure does not portray that outcome. The figure shows that immediately after completing the tutor, which required one accurate construction of the entire program, only two subjects (S2 & S5) were able to write the program correctly. After the subjects received classroom instruction and ran the Applet, eight of the 12 subjects were able to write the program correctly. Notably, however, on the final assessment occasion , which occurred 12 weeks later, only one subject (S3) was able to write the program correctly. These data show the improvement in performance over the first three repetitions of writing the program and the forgetting of the program after the 12-week delay interval.

Figue 6: Cumulative total correct programs written on the last three assessment occasions.

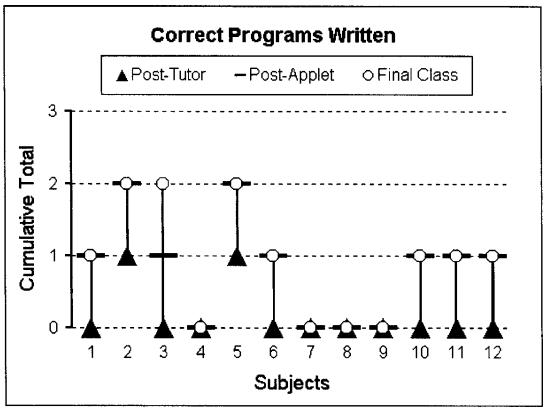
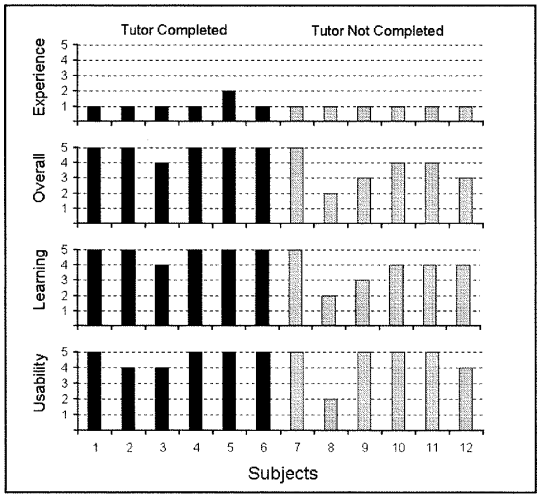


Figure 7 presents rating choices on the following four self-report scales: (1) prior experience with Java, (2) overall impression toward the tutor, (3) usefulness of the tutor in learning Java, and (4) usability of the interfaces. The data are grouped into those students who completed the tutor and those students who did not complete the tutor during the first classroom period. For Experience, only Subject 5 reported slight experience with Java prior to using the tutor. For Overall, five of the "completers" and one "non-completer" reported the maximum scale value. Although the values seem graphically lower for the non-completers, a Wilcoxon ranks test did not support a significant difference between the groups on this or any other scale (all $p > .20$). Similar effects were observed for Learning. For Usability, eight of the 12 subjects reported the highest scale value. It is also notable that Subject 8, who was in the "non-completers" group, gave consistently low ratings on these scales and also did not show accurate Applet construction on any of the three assessment occasions. Despite these reports, that subject did show an increase in reported confidence in the use of the Java items across the four assessment occasions. In summary, these self-report data support the conclusion that almost all subjects had positive reactions to the tutoring system and its methodology for programming a series of interactive instructional events.

Figure 7: Self-report data on four scales.



DISCUSSION

This study investigated a PSI for acquiring fundamental knowledge of a Java Applet. The instructional design of a Java tutoring system followed programmed instruction principles, which were supplemented with students' personal interactions with the course instructors and collaborations with peers. In contrast to passive online tutorials that only display information, no matter how skillfully organized and delivered within a hypermedia environment, the present tutoring system required learners actively to construct correct responses during the training. The level of complexity of the learned and constructed response was systematically increased until the final response was the production of the entire Java program. The programmed instruction component was augmented by a discussion with an "expert" that culminated in the learner's running of the Applet on the world-wide web. These factors together characterize a Personalized System of Instruction.

Although the data generally show confidence and performance improvements at least across the first three assessment occasions, it is notable, however, that only one student was able to write the Applet code correctly at the end of the course. This outcome presents a challenge for interpretation, especially since the course content throughout was the cumulative construction of an interactive information system using Java.

One way to understand this outcome, perhaps, is to consider the fact that the Java Applet class was taught and discussed only during the first two sessions of the course. The other components of the Java AWT that were taught over the remaining class periods did not involve repeating the initial Applet construction. It is also the case that the instructor emphasized to the students the importance of seeking and using online information from Sun Microsystems, Inc. about the construction and manipulation of the interface components that were presented in the course. In that sense, the students were not encouraged to memorize Java code. They were encouraged to seek out and use information as professionals. Furthermore, students were not required to write Java code on examinations, although they were required to interpret code fragments based on information presented in the course study guides. Nevertheless, these findings indicate the importance of repetition in achieving a fundamental competence that sets the occasion for future confidence and learning, and they show that students within a group may require different amounts of practice to achieve the identical level of skill. They also show that competence displayed on one occasion may not be easily retained.

Despite these observations, the Personalized System of Instruction (Keller, 1968), with the Java tutoring system as the central component, has been adopted by the authors to good advantage in the classroom because it generates a history of symbol use and program construction competence in each individual student. It combines both teaching and testing within a single conceptual framework: programmed instruction. It allows the needs of the individual student to be met because it frees the teacher from relying exclusively on traditional approaches, such as lecturing and writing on the board, to deliver technical information to a group of students. Most importantly, perhaps, the present tutoring system combines knowledge delivery with learning, assessment, and documentation of competence. Although the number of subjects in the present study was low, the concentration on the behavior of the individual learner, rather than on group averages, together with past and planned future replications with different class groups, enables the cumulative development of a knowledge base that will document the reliability of the current findings under conditions that promote the generality of the programmed instruction methodology.

Much has been written about the nature of computer programming and the attainment of expertise as a high-level problem-solving activity (e.g., Anderson, Corbett, Koedinger, & Pelletier, 1995). Our classroom experience, however, continues to indicate the importance of not overlooking basic learning parameters of guided rehearsal and correct practice. These latter conditions help make computer programming accessible to inexperienced and unconfident students who may otherwise withdraw from the initial effort required to achieve the background competence necessary to acquire advanced programming skills. Although it is certain that complex problem-solving skills and conceptual understanding are considerations in the development of expertise as a computer programmer, there is, perhaps, underestimated value to fostering an inductive development of these important outcomes by the simple repetition of fundamental response patterns.

REFERENCES

Anderson, J.R. (1995). *Learning and Memory: An Integrated Approach*. New York: Wiley.

Anderson, J.R., Corbett, A.T., Koedinger, K.R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of Learning Science*, *4*, 167-207.

Bloom, B.S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, *13*, 4-16.

Brock, J.F. (1997). Computer-based instruction. In G. Salvendy (Ed.), *Handbook of Human Factors and Ergonomics* (pp. 578-593). New York: Wiley.

Durham, A.G., & Emurian, H.H. (1998). Learning and retention with a menu and a command line interface. *Computers in Human Behavior*, *14*, 597–620.

Emurian, H.H., Hu, X., Wang, J. & Durham, A.G. (2000). Learning Java: A programmed instruction approach using Applets. *Computers in Human Behavior*, *16*, 395-422.

Ferster, C.B., & Perrott, M.C. (1968). *Behavior Principles*. New York: Appleton-Century-Crofts.

Greer, R.D., & McDonough, S.H. (1999). Is the learn unit a fundamental measure of pedagogy? *The Behavior Analyst*, *22*, 5-16.

Holland, J.G. (1960). Teaching machines: An application of principles from the laboratory. *Journal of the Experimental Analysis of Behavior*, *3*, 275-287.

Keller, F.S. (1968). Goodbye teacher... *Journal of Applied Behavior Analysis*, *1*, 79-89.

Kritch, K.M., & Bostow, D.E. (1998). Degree of constructed-response interaction in computer-based programmed instruction. *Journal of Applied Behavior Analysis*, *31*, 387–398.

Li, S., & Lewandowsky, S. (1995). Forward and backward recall: Different retrieval processes. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *21*, 837-847.

Reigeluth, C.M., & Darwexeh, A.N. (1982). The elaboration theory's procedures for designing instruction: A conceptual approach. *Journal of Instructional Development*, *5*, 22-32.

Skinner, B.F. (1958). Teaching machines, *Science*, *128*, 969-977.

Tennyson, R.D., & Schott, F. (1997). Instructional design theory, research, and models. In R.D. Tennyson, F. Schott, N.M. Seel, & S. Dijkstra (Eds.), *Instructional Design: International Perspectives* (pp. 1-16), Mahwah, NJ: Lawrence Erlbaum Associates.