Teaching Java™: Managing Instructional Tactics to Optimize Student Learning

Henry H. Emurian, Information Systems Department, UMBC, 1000 Hilltop Circle, Baltimore, MD 21250, USA; E-mail: emurian@umbc.edu

INTRODUCTION

Direct mastery of the core knowledge in a discipline is increasingly recognized as a fundamental requirement to applying and extending that knowledge to solve novel problems. That recognition implies an instructional design to overcome the empirically verified shortcomings of teaching tactics that provide minimal guidance during a student's learning experiences (Kirschner, Sweller, & Clark, 2006). In that regard, our previous work consistently confirmed the value of programmed instruction in teaching introductory Information Systems students a Java applet as a first technical training exercise in preparation for advanced learning (Emurian, 2004, 2005, 2006a,b). Similar value of programmed instruction is evident in its applications within other disciplines, such as chemistry (Kurbanoglu, Taskesenligil & Sozbilir, 2006). The objectives of our work are to apply programmed instruction and to assess its effectiveness as a tactic to promote a common level of mastery by all students for a designated learning objective in Java programming. An optimal level of mastery is taken to reflect a *true gain* in learning (Anderson, Corbett, Koedinger, & Pelletier, 1995).

Among several recommendations for effective learning principles to promote retention and transfer of knowledge, however, are repeated practice with different instructional modalities (Halpern & Hakel, 2003) and socially supported interactions (Fox & Hackerman, 2003). The modalities that have been adopted in our classroom applications include (1) programmed instruction, (2) lectures with hands-on learning, and (3) collaborative peer tutoring. Although these tactics are demonstrably effective in promoting programming skill, software self-efficacy, and generalizable knowledge, our most recent assessment of learning effectiveness showed room for improvement in the goal of achieving maximal learning in all students on tests of far transfer following the collaborative peer tutoring (Emurian, 2006b). To potentiate the effectiveness of the collaborative peer tutoring, then, the present evaluation was undertaken with a modification to the instructions and materials that were presented to students to prepare for peer tutoring and to use during the collaboration session. The procedure also allowed the collaborating students to view and discuss together the questions that constituted the tests of far transfer. Finally, the Java program to be learned by students as the first technical exercise was updated to Java swing, and it contained more items to be mastered in comparison to the previous work in this area of classroom applications and research.

METHOD

Subjects

Subjects were 13 graduate students, four females and nine males, taking IS 613 (*GUI Systems Using Java*) during a four-week summer session (Summer 2006). The class met three times each week, and each class lasted three hours. The course was designed for Information Systems students, and the prerequisite was one prior programming course.

The background characteristics of the students were as follows: age (median = 28 years, range = 23 to 33), number of prior programming courses taken (median = 3, range = 1 to 15), rated prior Java experience (median = 2, range = 1 to 5 on a 10-point scale presented below), and rated prior programming experience (median = 5, range = 2 to 8 on a 10-point scale presented below).

The research protocol was exempt from informed consent by the Institutional Review Board, and the course syllabus clearly indicated that questions both embedded in the Java tutor and administered during several assessment occasions in class were eligible to appear on a quiz. The course description and syllabus provided information about the Java tutor and the collaborative peer tutoring, and they presented the rationale for the repetition of initial learning using the several different instructional modalities under consideration.

Material

Java Program

The instructional tactics in this study were based upon teaching students a JApplet program that would display a JLabel object within a browser window. The program was arbitrarily organized into 11 lines of code (*e.g.*, ***JLabel myLabel;***) and 37 separate items of code (*e.g.*, ***getContentPane()***). The 11 lines of code are as follows:

(1) **import javax.swing.JApplet;**
(2) **import javax.swing.JLabel;**
(3) **import java.awt.Color;**
(4) **public class MyProgram extends JApplet {**
(5) **JLabel myLabel;**
(6) **public void init() {**
(7) **myLabel = new JLabel("This is my first program.");**
(8) **getContentPane().setBackground(Color.yellow);**
(9) **getContentPane().add(myLabel);**
(10)**}**
(11)**}**

Access to the web-based Java tutor, as presented below, will also show the complete program as part of the tutor's instructions to the student.

Questionnaires[1]

Java software self-efficacy was assessed by requesting a rating of confidence, for each of the 23 unique items of code (*e.g.*, ***import***) in the program, in being able to use the Java code to write a program that displays a text string, as a JLabel object, in a browser window. The scale anchors were ***1 = No confidence.*** to ***10 = Total confidence.*** Twelve multiple-choice questions were administered that required applying a general concept of Java object-oriented programming to solve. These questions did not appear within the Java tutor, and they were intended to assess far transfer or meaningful learning (Mayer, 2002). Each question had five choices, and for each question, a rating of confidence was made that the selected choice was the correct choice. The scale anchors were ***1 = Not at all confident.*** to ***10 = Totally confident.*** Ratings of classification and functionality learning for eight Java identifiers were also obtained, but they are beyond the scope of this paper.

The pre-tutor questionnaire also solicited demographic information, to include age, sex, and college major. The total number of prior programming courses taken was also requested. Two programming experience rating scales were presented, one for general programming experience and one for Java programming experience. For both scales, the anchors were ***1 = No experience. I am a novice.*** to ***10 = Extensive experience. I am an expert.***

The post-tutor questionnaire omitted the demographic information, and it assessed evaluations of the tutor for (1) overall effectiveness, (2) effectiveness in learning Java, and (3) usability. The anchors were ***1 = Lowest value.*** to ***10 = Highest value***.

Procedure

Java Tutor

At the first class meeting, students completed the pre-tutor questionnaire. Students next completed the web-based Java tutor[2]. The tutor taught a JApplet that displays a text string, as a JLabel object, in a browser window on the web. The Java code and a brief description of the eight stages of the tutor are presented as part of the open source material[3]. When a student finished the tutor, he or she next completed a post-tutor questionnaire, which duplicated the software self-efficacy ratings and multiple-choice rules questions and confidence ratings. The student next accessed a set of questions and guidelines, which were posted on Blackboard, that were to be used to structure the collaborative peer tutoring session during a subsequent class. This material also presented a link to access the textual explanations of the items and lines of code that were presented in the Java tutor. The instructions with this material indicated that the questions presented were eligible to appear on a quiz.

Lecture

At the second class meeting, the author gave a lecture on the program taught in the Java tutor. The students wrote the code in a Unix™ text editor during the lecture, which repeated the information presented in the tutor. The students were also taught the HTML file, used to access the Java bytecode file, as a URL on the web. Support was provided so that all students successfully ran the JApplet program at the conclusion of this lecture-based exercise.

This lecture required approximately one hour to complete, and the remaining class time was spent on the next unit of material, which related to the life cycle of an Applet. Students were encouraged to help each other during the subsequent classes in the semester, which combined lectures and hands-on demonstrations, with the understanding that files were not to be copied without prior permission of the instructor.

Interteaching

At the third class meeting, a collaborative peer tutoring session occurred based upon the dyadic "interteaching" model (Boyce & Hineline, 2002). Students formed six groups of two and one group of three students for the session, which lasted one hour. The assignment was for the students to discuss the set of questions and guidelines made available at the conclusion of the Java tutor work undertaken at the first class meeting. Also presented was the questionnaire, and students were encouraged to discuss the items together prior to answering individually. This was the major innovation in the study, providing the opportunity for students to discuss the rules questions together. The interteaching questionnaire instructions stated that the 12 rules questions were eligible to appear on a quiz, but the remaining items were there only to assess instructional effectiveness of the interteaching session. The interteaching questionnaire also requested ratings of the effectiveness of the session for (1) learning the material and (2) readiness to be tested on the material, where ***1 = Not effective*** to ***10 = Totally effective***.

During the interteaching session, students posted questions on a Blackboard discussion board, and the instructor provided feedback. For the 12 rules questions, the correct selection was never given. Instead, the instructor responded in a way that made certain that students understood the general principle underlying the correct choice, and this process was occasionally iterative.

On the same day as the interteaching session, the instructor posted an announcement on Blackboard giving a rules question that was answered incorrectly by two of the students. The announcement was as follows: "Some students answered 'c' below for this question [also presented in the announcement]. The 'c' choice is not correct because JScrollPane is a class, not an object. An object name begins with a lowercase letter. If you have a question about this, please send me email." All student inquiries were answered privately in a way to promote understanding of the principle involved. The correct answer was not given.

Graded Quiz

At the fourth class meeting, a quiz was administered that included questions embedded within the Java tutor and the 12 rules questions as indicated above. The graded quiz did not include any rating assessments.

RESULTS

Figure 1 presents boxplots of correct answers on the rules test over the five assessment occasions. For each of the 12 questions answered during the Pre-Tutor assessment, one student did not select any answer, but instead indicated being unprepared to answer. The figure shows graphically that the median total correct answers increased over the first four occasions and reached the ceiling of 12 on the Interteaching occasion. A Friedman test (Conover, 1971, p. 264) was significant (Chi-Square = 42.259, df = 4, p = 0.000). The figure also shows that the greatest change occurred between the Pre-Tutor and Post-Tutor occasions, and both medians were 12 for the Interteaching and Quiz occasions. A Welch robust test (Maxwell & Delaney, 2004, p. 134), based on the differences, Di, in correct answers between successive pairs of occasions over the five occasions, was significant (W = 10.889, p = 0.000). Planned pairwise comparisons were significant[4] for D1 compared to D2 (W=10.145, p = 0.005), not significant for D2 compared to D3 (W = 1.513, p = 0.231), and significant for D3 compared to D4 (W = 12.295, p = 0.003).

Figure 1. Correct answers on rules test

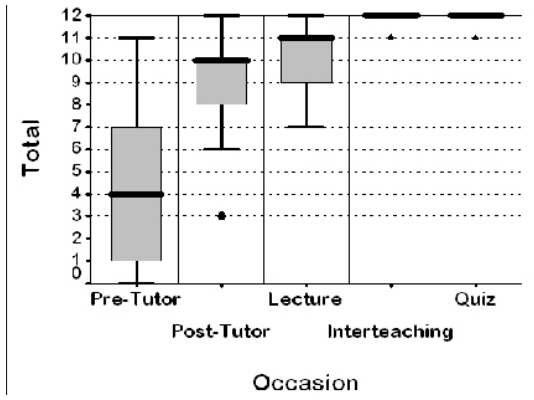


Figure 2 presents boxplots, over four successive occasions, of the ratings made by the students regarding confidence that the selected answer on the rules test was correct for answers that were Right and for answers that were Wrong. Ratings were not obtained during the graded quiz. The number below each boxplot reflects the number of students who answered Right and/or Wrong over the four assessment occasions, and that is the reason that the frequency for a boxplot is sometimes less than 13 (*e.g.*, number of students giving incorrect answers for the interteaching occasion). The Welch robust test, used because of unequal sample sizes, was significant for Right answers (W = 16.632, p = 0.000) and for Wrong answers (W = 40.864, p = 0.000). The latter test was based on the first three occasions because the variance for the Interteaching occasion was zero. For Right answers, planned pairwise comparisons were significant for Pre-Tutor and Post-Tutor (W = 27.398, p = 0.000), not significant for Post-Tutor and Lecture (W = 0.108, p = 0.745), and not significant for Lecture and Interteaching (W = 4.959, p = 0.044) occasions. For Wrong answers, planned pairwise comparisons were significant for Pre-Tutor and Post-Tutor (W = 55.646, p = 0.000) and not significant for Post-Tutor and Lecture (W = 1.220, p = 0.282) occasions. An overall comparison of confidence ratings between Right and Wrong answers was significant (W =

Figure 2. Confidence in accuracy of rules test answers

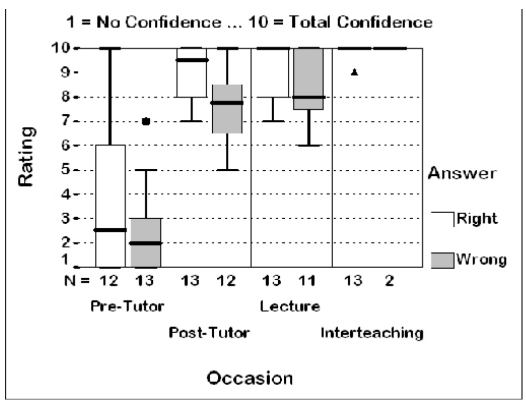
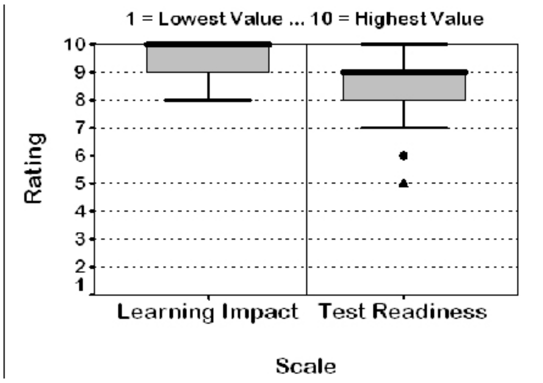


Figure 3. Interteaching evaluation



9.481, p = 0.003). Confidence generally increased over the assessment occasions, reaching the ceiling for correct answers after the lecture. However, confidence was seen to increase for both correct and incorrect answers, although an overall comparison favored the correct answer choices.

Figure 3 presents boxplots of ratings on the interteaching evaluation, which was administered at the conclusion of the interteaching session. The figure shows graphically the students' reported value in the interteaching session even when it occurred after using the Java tutor and after running the program on the web. The median rating of learning impact reached the scale's ceiling of ten, with eight being the lowest rating observed. The rating of test readiness was only slightly less, with a median of nine. A Friedman's test was significant (Chi-Square = 5.444, p = 0.020). Similar to our previous work, the ratings of test readiness were lower than corresponding ratings of learning impact. These show that the students reported value in the collaborative peer tutoring even when the session followed several other instructional experiences.

Figure 4 presents boxplots of software self-efficacy ratings across the first four assessment occasions. These ratings were not obtained during the graded quiz. Each boxplot is based upon the median rating over the 23 unique items of code in the program for the 13 students. Cronbach's alpha reliability of the ratings within each assessment exceeded 0.90, and all were significant (p < .05). A Friedman test was significant (Chi-Square = 32.614, df = 3, p = 000). A Welch test, based on the differences in correct answers between successive pairs of occasions, was significant (W = 30.222, p = 0.000). Planned pairwise comparisons of the differences, Di, were significant for D1 compared to D2 (W = 60.215, p = 0.000) and not significant for D2 compared to D3 (W = 1.330, p = 0.260). Software self-efficacy increased over the assessment occasions, and it reached the ceiling following the lecture.

Figure 5 presents boxplots of ratings of evaluation of the tutor taken during the Post-Tutor assessment. Medians for all three scales reached the scale ceiling of ten, with only a single outlier observed for Java Learning. These data show that students reported value in their use of the tutor.

Figure 4. Software self-efficacy

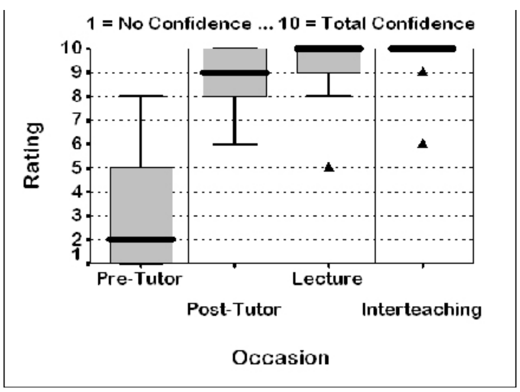
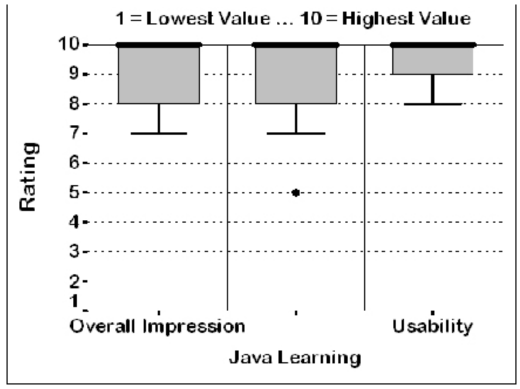


Figure 5. Evaluation of the tutor



DISCUSSION

The results of this study show the value of applying several different instructional modalities in furtherance of having Information Systems students achieve a common level of skill and understanding in a simple Java applet, presented as a first technical exercise in a semester-long course. The data support the utility of this approach as reflected in students' rules test performance and software self-efficacy, which progressively improved over the successive assessment occasions. Rehearsal is an intuitively obvious and well-researched factor in knowledge and skill acquisition (*e.g.*, Salas & Cannon-Bowers, 2001), and the present study shows how structured rehearsal may be managed using the several modalities under consideration. Principles underlying such managed skill acquisition with different instructional modalities are presented elsewhere (Fox & Hackerman, 2003; Halpern & Hakel, 2003).

Having students discuss rules questions together enhanced understanding in the present context. Similar to our previous observations, however, students showed "overconfidence" in incorrect rules answers, and that issue requires exploration in the design of future work. Importantly, students reported value in the Java tutor and in the collaborative peer tutoring, and taken together with the lecture, these approaches to managing rehearsal in the classroom environment converge on what are increasingly recognized as vital ingredients to facilitate science education, in general (DeHaan, 2005).

This study constitutes a systematic replication (Sidman, 1960) of a set of teaching tactics that were revised with the expectation that student learning would be improved as a consequence. The methodology reflects design-based research, which is a type of formative evaluation (Collins, Joseph, & Bielaczyc, 2004) that is emerging as an alternative methodology in support of developing and assessing improvements in instructional design within the context of the classroom (Bell, Hoadley, & Linn, 2004; Design-Based Research Collective, 2003). In that regard, the order of presenting the several instructional tactics was determined by anecdotal observations of student performance over the several classroom evaluations that were previously undertaken in this stream of work. It was decided that a hands-on lecture would benefit from students' prior rehearsal with the Java code and that collaborative peer tutoring would benefit from the cumulative learning obtained from the programmed instruction and the lecture. Since the components in the current ordering are well received by students and since a desired learning outcome was achieved, we have the view that it is worthwhile now to direct our attention to developing advanced instructional material, rather than to "prove" the optimal ordering under conditions of a traditional "effect-size" experiment. Support for that view is implicit within designed-based research and has been discussed by educational scholars (*e.g.*, Mayer, 2004; Sackett & Mullen, 1993).

There are many approaches to teaching computer programming, ranging from an emphasis on mathematics and algorithms (Hu, 2006) to supportive programming environments such as BlueJ (Kolling, Quig, & Rosenberg, 2003), DrJava (Hsia, Simpson, Smith, & Cartwright, 2005), Problem-Based Learning (Tsang & Chan, 2004), PigWorld (Lister, 2004), and the Environment for Learning to Program

(Truong, Bancroft, & Roe, 2005).The instructional tactics adopted here in the classroom at the start of a semester's work are based initially upon *programmed instruction*, which is a form of structured and optionally automated instruction, as discussed by Emurian and Durham (2003) and Emurian, Wang, and Durham (2003) with respect to teaching computer programming. They also include *interteaching*, which is a form of collaborative peer tutoring (Boyce & Hineline, 2002). As implemented in the present context, these tactics originated from behavior analysis, and the Cambridge Center for Behavioral Studies[5] provides fundamental definitions and a wealth of information regarding the philosophical underpinnings and applications of this approach to science, in general, and to education, in particular. Finally, these tactics are to be understood as providing only an initial series of learning experiences to students in preparation for subsequent learning with other instructional and program development tools and techniques, to include the use of an integrated development environment (IDE) such as Eclipse.

Behavior analysis is one promising approach in identifying the ontogenetic instructional learn units (Greer & McDonough, 1999) whose mastery provides the textual tools essential for advanced understanding, thinking, and problem solving in the domain of computer programming and beyond (Skinner, 1957). Teachers facing the difficult challenge of providing effective instruction to the diversity of students who enroll in introductory computer programming courses need to be mindful of all approaches to helping their students succeed. The present study represents one set of instructional tactics that are demonstrably effective for Information Systems students.

REFERENCES

Anderson, J.R., Corbett, A.T., Koedinger, K.R., & Pelletier, R. (1995). Cognitive tutors: lessons learned. *Journal of the Learning Sciences*, *4*(2), 167-207.

Bell, P., Hoadley, C.M., & Linn, M.C. (2004). Design-based research in education. In M.C. Linn, E.A. Davis, & P. Bell (Eds.). *Internet Environments for Science Education* (pp. 73-88), Laurence Erlbaum Associates.

Boyce, T.E., & Hineline, P.N. (2002). Interteaching: a strategy for enhancing the user-friendliness of behavioral arrangements in the college classroom. *The Behavior Analyst*, *25*, 215-226.

Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: theoretical and methodological issues. *Journal of the Learning Sciences*, *13*(1), 15-42.

Conover, W.J. (1971). *Practical Nonparametric Statistics.* New York, NY: John Wiley & Sons, Inc.

DeHaan, R.L. (2005). The impending revolution in undergraduate science education. *Journal of Science Education and Technology*, *14*(2), 253-269.

Design-Based Research Collective. (2003). *Educational Researcher*, *32*(1), 5-8.

Emurian, H.H. (2004). A programmed instruction tutoring system for Java: consideration of learning performance and software self-efficacy. *Computers in Human Behavior*, *20*(3), 423-459.

Emurian, H.H. (2005). Web-based programmed instruction: evidence of rule-governed learning. *Computers in Human Behavior*, *21*(6), 893-915.

Emurian, H.H. (2006a). A web-based tutor for Java™: evidence of meaningful learning. *Journal of Distance Education Technologies*, *4*(2), 10-30.

Emurian, H.H. (2006b). Assessing the effectiveness of programmed instruction and collaborative peer tutoring in teaching Java™. *International Journal of Information and Communication Technology Education*, *2*(2), 1-16.

Emurian, H.H., & Durham, A.G. (2003). Computer-based tutoring systems: a behavioral approach. In J.A. Jacko and A. Sears (Eds.), *Handbook of Human-Computer Interaction* (pp. 677-697). Mahwah, NJ: Lawrence Erlbaum & Associates.

Emurian, H.H., Wang, J., & Durham, A.G. (2003). Analysis of learner performance on a tutoring system for Java. In T. McGill (Ed.), *Current Issues in IT Education* (pp. 46-76). Hershey, PA: IRM Press.

Fox, M.A., & Hackerman, N. (2003). *Evaluating and improving undergraduate teaching in science, technology, engineering, and mathematics.* Washington, DC: The National Academies of Science Press.

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

Halpern, D.F., & Hakel, M.F. (2003). Applying the science of learning to the university and beyond: teaching for long-term retention and transfer. *Change*, *35*(4), 37-41.

Hsia, J.I., Simpson, E., Smith, D., & Cartwright, R. (2005). Taming Java for the classroom. *SIGCSE'05,* February 23-27, St. Louis, MI, 327-331.

Hu, C. (2006). It's mathematical after all: the nature of learning computer programming. *Education and Information Technologies*, *11*(1), 83-92.

Kirschner, P.A., Sweller, J., & Clark, R.E. (2006). Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, *41*(2), 75-86.

Kolling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Journal of Computer Science Education*, *14*(Dec), 1-12.

Kurbanoglu, N.I., Taskesenligil, Y., & Sozbilir, M. (2006). Programmed instruction revisited: a study on teaching stereochemistry. *Chemistry Education Research and Practice*, *7*(1), 13-21.

Lister, R. (2004). Teaching Java first: experiments with a pigs-early pedagogy. *Proceedings of the Sixth Conference on Australian Computing Education* (pp. 177-183), Volume 30, Dunedin: Australian Computer Society, Inc.

Mayer, R.E. (2002). *The promise of educational psychology. Volume II. Teaching for meaningful learning*. Upper Saddle River, NJ: Pearson Education, Inc.

Mayer, R.E. (2004). Should there be a three-strikes rule against pure discovery learning? *American Psychologist*, *59*(1), 14-19.

Maxwell, S.E., & Delaney, H.D. (2004). *Designing Experiments and Analyzing Data: A Model Comparison Perspective. Second Edition*. Mahwah, NJ: Lawrence Erlbaum Associates.

Sackett, R.R., & Mullen, E.J. (1993). Beyond formal experimental design: towards an expanded view of the training evaluation process. *Personnel Psychology*, *46*, 613-627.

Salas, E., & Cannon-Bowers, J.A. (2001). The science of training: a decade of progress. *Annual Review of Psychology*, *52*, 471-499.

Sidman, M. (1960). *Tactics of Scientific Research*. New York: Basic Books.

Skinner, B.F. (1957). *Verbal Behavior*. New York: Appleton-Century-Crofts, Inc.

Truong, N., Bancroft, P., & Roe, P. (2005). Learning to program through the web. *Proceedings of the 10th annual SIGSCE conference on innovation and technology in computer science education*, ITiCSE'05, June 27–29, Monte de Caparica, Portugal. ACM Press.

Tsang, A.C.W., & Chan, N. (2004). An online problem-based model for the learning of Java. *Journal of Electronic Commerce in Organizations*, *2*(2), 55-64.

ENDNOTES

1. The Java tutor source code and all assessment instruments, to include the rules test and quiz, are freely available on the web: http://nasa1.ifsm.umbc.edu/irma/2007/
2. The Java tutor is freely accessible on web, and this report is based on version 2 of the tutor. The course material is also freely available: http://nasa1.ifsm.umbc.edu/IFSM413_613/
3. http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/TutorLinks.html
4. To control for the experimentwise error rate, the significant *p* value for each planned comparison must be less than 0.05/number-of-planned-comparisons.
5. http://www.behavior.org/index.cfm