IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4878

# Web-Based Tutoring for Java™: Edvidence of Rule-Governed Learning

Henry H. Emurian
Information Systems Department, ITE 420, UMBC, 1000 Hilltop Circle, Baltimore, Maryland 21250 USA, emurian@umbc.edu

## INTRODUCTION

This paper presents a continuation of our work to develop and refine an automated instructional program to assist information systems students in beginning their study of Java™ We have previously reported our progress in the development of this tutoring system and its application as the first technical exercise for students in a programming course (Emurian & Durham, 2001, 2002; Emurian, Hu, Wang, & Durham, 2000). The purpose of the system is to provide each and every student with a documented and identical level of elementary knowledge and skill. The tutoring system has been demonstrably effective in promoting skill and confidence in students by giving them a successful learning experience that motivates their further study of Java using textbooks, lectures, laboratory demonstrations, and the like.

One of the challenges in developing an automated instructional system, however, is to craft the learning experience so that students acquire the capability to solve problems not explicitly taught or encountered in the system itself. When students are able to apply knowledge successfully to new situations, they are said to be demonstrating *meaningful learning* (Mayer, 2002) as opposed to reciting facts acquired by rote memorization. These two outcomes reflect the opposite endpoints on a generality-specificity dimension of skill (e.g., Novick, 1990). Generalizable rules, which may be the essence of meaningful learning, can be acquired by direct instruction and rehearsal or by induction, when many different situations are encountered that exhibit the general rule (e.g., Kudadjie-Gyamfi & Rachlin, 2002). The former tactic is consistent with our instructional system design, which is competency based and which is intended to insure that all students reach the same level of knowledge and skill. The purpose of the present study, then, is to show that students who complete the tutor do acquire general rules that are applicable to problems not explicitly addressed in the tutor itself.

Interpretative surveys of the scientific literature in *far transfer* effects of learning continue to show the advantages of explicitly teaching generalizable principles and rules, rather than expecting such knowledge to develop implicitly or abstractly as a by-product of memorizing facts (Barnett & Ceci, 2002). For example, it is likely more efficient to teach students the rule to begin the name of a Java class with a capital letter rather than have students acquire such a rule inductively by memorizing many different programs and trying to discover commonalities. In fact, a combination of teaching rules with examples might be optimal for meaningful learning, and our approach to the design of the tutoring system is based on the latter assumption.

## METHOD

### Subjects

The subjects were eight female and four male graduate students in information systems enrolled in a course (Summer 2003) entitled *Graphical User Interface Systems Using Java*. The median age was 27 years (range = 21 to 49 years). On a ten-point ordinal scale, where *1 = novice* and *10 = expert*, the median prior experience with Java was 1 (range = 1 to 4). The median number of programming courses taken previously was 3 (range = 1 to 5).

### Materials

The tutoring system teaches a simple Java Applet, which is a program that is downloaded from a server and run in a browser. It is intended for students who are not proficient computer programmers and who may lack confidence in their ability to write a program that works. The Applet program is organized into 32 items and ten rows. The student is taught the meaning of each item and the meaning of each row. The student must pass a multiple-choice test on these program components, and studying continues until each test is passed correctly.

The seven stages of the tutoring system, from basic instructions and code examples to the construction of the code from memory, are presented in Emurian (in press) and Emurian, Wang, and Durham (2003). In brief, the tutoring system is an interactive system that combines teaching, assessment of competency, and rehearsal within a single framework. The design of the system is based upon programmed instruction, which takes a learner through a series of experiences from simple mastery of the form of symbols to writing and understanding a complete program. This design reflects the application of behavior analysis principles to designing teaching strategies for technology education (Greer, 2002, p. 185).

Information is delivered to the student in a *frame*. A frame consists of (1) the presentation of the symbol to be learned, within the proper context; (2) a textual display of information about the symbol's meaning and use; (3) a multiple-choice test on the meaning of the symbol; and (4) an input field for typing the symbol by recall. If the student makes an error during steps 3 or 4, the tutor resets to step 1. These elements of a frame constitute a *learn unit* (Greer & McDonough, 1999).

Below is presented the ninth of the 32 item frames in the tutoring system. It teaches the meaning of the *MyProgram* item, which is a subclass of the Applet class in the program.

**MyProgram frame:**

The term *MyProgram* is the name of the class that you are writing. Your Java program is a class. The name is an arbitrary alphanumeric string. *MyProgram* is not the name of an instance of this class. It is the name of the class. It is important that you begin to distinguish the name of a class from the names of particular instances of that class that are created later. This distinction will become clearer as you progress through the tutor. Notice that the name of the class begins with a capital letter. That is a convention in Java. The name of a class begins with a capital letter. That is an important rule to know.

The text file that contains the Java program for the *MyProgram* class must have exactly the same name, together with "dot java" at the

end. The file for your program would be named *MyProgram.java*. The name of the text file must exactly match the name of the class. That is an important rule to know.

The Java text file, which is the source program, will be compiled with: *javac MyProgram.java*. The result of compiling the program is a class file named *MyProgram.class*, which will be located in your directory.

Below are the five rule-based multiple-choice questions. The correct solution to each question requires the application of a general principle that was presented in the frames of information. None of the items below appeared in the frames or in the multiple-choice tests that were embedded in the frames. This eliminated rote memorization as an explanation for correct solutions, should they be observed at all. For each of the five rule-based questions, the student rated his or her confidence that the correct answer was selected. The ordinal scale anchors were 1 and 10, where *1 = no confidence* and *10 = total confidence*. A ten-point rating scale was adopted to increase the sensitivity of the scale to detect changes in ratings over three successive assessment occasions.

**Rule-Based Questions:**

1. Which of the following lines most likely would be used to create a shorthand notation for the Frame class, which is built in to Java?
   a. import java.awt.frame;
   b. import java.awt.Frame.class;
   c. import java.awt.Frame;
   d. import java.awt.frame.class;
2. Which of the following lines most likely would be used to construct an instance of a Button class?
   a. MyButton = new Button("Hello");
   b. myButton = new Button("Hello");
   c. myButton = button.class("Hello");
   d. MyButton = Button("Hello");
3. Which of the following lines most likely would be used to add a Checkbox object to a container?
   a. Add(myCheckBox);
   b. Add(Checkbox);
   c. add(Checkbox);
   d. add(myCheckBox);
4. Which of the following lines most likely overrides a method that is contained in the Applet class?
   a. public void stop(){ lines of Java code here }
   b. public void Stop{} { lines of Java code here }
   c. Public void Stop() ( lines of Java code here )
   d. Public void stop() { lines of Java code here }
5. Which of the following sequences is correct?
   a. declare a TextField object, construct a TextField object, add a TextField object to a container
   b. construct a TextField object, declare a TextField object, add a TextField object to a container
   c. declare a TextField object, add a TextField object to a container, construct a TextField object
   d. add a TextField object to a container, declare a TextField object, construct a TextField object

**Procedure**

At the first class meeting, students used the web-based tutor1. All students completed all stages in the tutor within the 3-hr class period. This means that all students left the first class period being able to write the ten lines of Java code from memory and with no error. The students had also studied the frames until they could accurately answer all multiple-choice test questions accurately.

Prior to using the tutor, students completed a questionnaire that obtained demographic information. For each of the 21 unique items of code in the program, the student rated his or her confidence in being able to use the symbol, where *1 = no confidence* and *10 = complete confidence*. This was the measure of software self-efficacy, based on the original work by Bandura (1977) and the later adoption of this approach by researchers in information technology education (e.g., Potosky, 2002; Torkzadeh & Van Dyke, 2002). Students also completed the above five rule-based questions, to include the rating of confidence in the accuracy of the answer selected.

After the students completed the tutor, they repeated the software self-efficacy and rule-based questions. The students then rated the overall quality of the tutor, the effectiveness of the tutor in learning Java, and the usability of the tutor interfaces. Each of these latter scales was a 10-point ordinal scale, where *1 = poor quality* and 10 = *best quality*.

During the second class, which was two days later, the author repeated the teaching of the Applet, but this time a lecture and discussion format was used. The author wrote the program on the board and discussed each item and row. The students simultaneously entered the program into a Unix™text editor. At the completion of the lecture, the students were taught how to compile the program into byte code. Additionally, the Unix directory tree and file protections were taught. The HTML file was then taught, and the students ran the Applet on the world wide web. Finally, the students repeated the questionnaire assessing software self-efficacy and rule-based learning.

## RESULTS AND DISCUSSION

Figure 1 presents boxplots of total correct rule-based answers by all students across the three assessment occasions: pre-tutor, post-tutor, and post-applet. The figure shows graphically that the median value increased over the three occasions, and a Kruskal-Wallis test was significant (chi-square = 16.93, df = 2, $p < .001$). The figure also shows that the most pronounced increase occurred between the pre-tutor and post-tutor occasions, in comparison to the post-tutor and post-applet occasions. A comparison of the means of the differences, for all 12 subjects, between pre-tutor and post-tutor totals (Mean = 2.3) with post-tutor and post-applet totals (Mean = 0.3) was significant, t(17.5) for unequal variances = 4.24, $p < .001$.

Figure 2 presents boxplots of median confidence ratings for Right (R) and Wrong (W) answers for pre-tutor (Pre-T), post-tutor (Post-T), and post-applet (Post-A) assessment occasions. The total number of medians within each occasion exceeds 12 because students often made both correct and incorrect choices for the answers. Values in the boxplots are based upon the collection of median ratings for each student for R and W answers across the three occasions. Comparisons between R and W medians within pre-tutor and post-tutor occasions were not significant. Accordingly, data were combined for R and W within each occasion, and a Kruskal-Wallis comparison between pre-tutor and post-tutor ratings was significant (chi-square = 18.56, $p < .001$). Too few medians were present in the W category for the post-applet assessment for a meaningful comparison using those data.

*Figure 1*

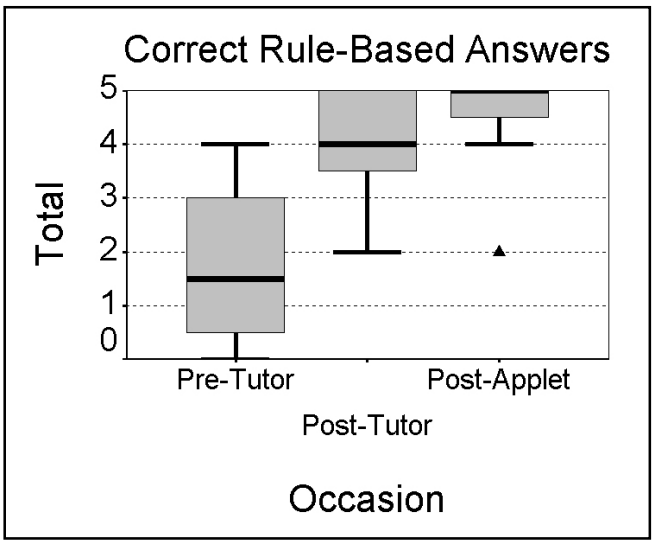


*Figure 2*

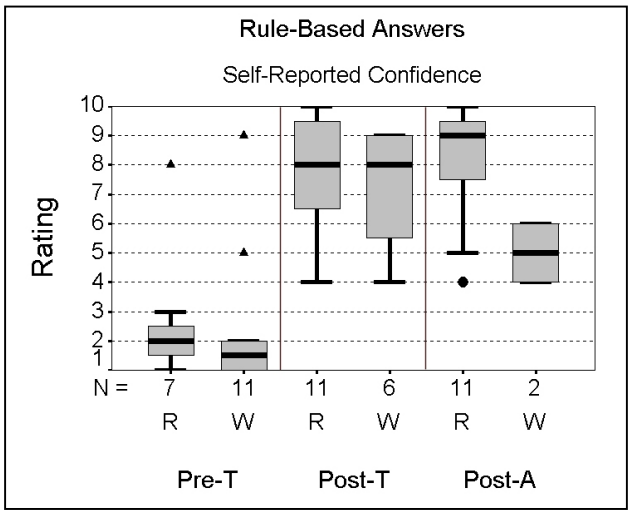
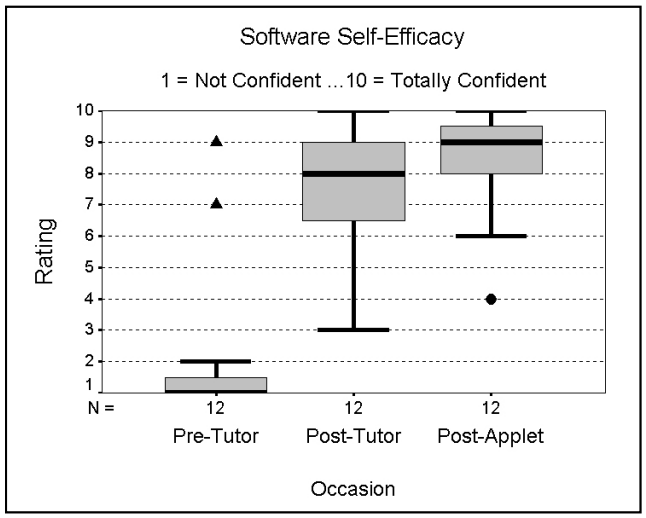



*Figure 3*




These ratings show the students' insensitivity in monitoring their own learning. Since the null hypothesis of no difference in confidence ratings between R and W answers could not be rejected, the learners did not know, perhaps, that their learning was incomplete. Since self-regulation of learning is an important skill (Veenman, Prins, & Elshour, 2002; Young, 1996; Zimmerman, 1994), how to achieve this outcome within the context of the present tutoring system warrants consideration as this teaching technology continues to mature.

Figure 3 presents boxplots of self-reports of software self-efficacy by all students across the three assessment occasions: pre-tutor (Cronbach's alpha = .97), post-tutor (Cronbach's alpha = .98), and post-applet (Cronbach's alpha = .98). The figure shows graphically that the median value increased over the three occasions, and a Kruskal-Wallis test was significant (chi-square = 18.45, df = 2, $p < .001$). The figure also shows that the most pronounced increase occurred between the pre-tutor and post-tutor occasions, in comparison to the post-tutor and post-applet occasions. A comparison of the means of the differences, for all 12 subjects, between pre-tutor and post-tutor ratings (Mean = 5.2) with post-tutor and post-applet ratings (Mean = 1.0) was significant, t(17.3) for unequal variances = 4.80, $p < .001$.

The post-tutor ratings of the tutoring system were as follows: median overall rating = 10 (range = 5 - 10); median effectiveness in teaching Java = 9 (range = 5 - 10); and median interface usability = 9.5 (range = 6 - 10). These outcomes are similar to the ratings observed in our previous studies, and they show that the tutoring system was generally well received by this population of students.

The importance of these self-ratings is to be understood in terms of the impact of the learning experience on the students' motivation to continue their studies. Given the frequently expressed concern that women and minority groups avoid science, mathematics, and engineering disciplines (Emurian, in press), it is encouraging to observe that at least some instructional tactics, such as programmed instruction, may be helpful to provide both skill and confidence in students who are initially interested in a discipline, but whose lack of preparation may result in demoralization to the point of avoiding or withdrawing from continued study. The essence of effective automated tutoring, then, is to provide a set of experiences that gives all students the skill and confidence to manage their own learning effectively, without regard to content and without the continued support of a tutoring system. Fostering such constructive metacognitive processes and supporting individual differences in ability constitute the foundation of effective automated tutoring (Cuevas, Fiore, Bowers, & Salas, in press).

As discussed elsewhere (Emurian & Durham, 2003), much of the literature in teaching computer programming addresses this matter as though the skill of computer programming requires a unique teaching technology. It is also directed toward groups of students rather than to the individual learner. Our approach is different. We assume that learning to write programs falls within the scope of training in general (Salas & Cannon-Bowers, 2001) and rule-governed learning in particular (Hayes, 1989). We also assume that a teaching technology can only be rationally developed and applied when it is directed toward the achievement of a criterion of mastery by each and every student. Research and interventions that are based on null hypothesis refutations of average performance between groups by definition accept at least some deficient student performance as an outcome. It is encouraging, then, to see the emergence of more achievement oriented research, based on pre-training and post-training comparisons in one group of learners, in contrast to between group comparisons. As indicated by Sackett and Mullen (1993), it may be more important to an organization to know that an instructional intervention will be successful for all learners than it is to know that mean performance between groups show differences.

It is an unfortunate irony that educational research is sometimes perceived as less valuable than other areas of research, at least within the social sciences (Sternberg & Lyon, 2002). The irony comes from the obvious importance of education and of knowing and applying the conditions that will help students to learn best, throughout the life span. In that regard, research in how best to teach computer programming is typically characterized by comparing average performance among several groups of learners, where each group is exposed to a somewhat different instructional condition. The literature is filled with scores of such studies. This strategy is inherently flawed because it tacitly accepts the outcome that not all students will achieve mastery even in the group with the best average performance.

The motivation for the above strategy, of course, is to determine the best instructional tactic, but that strategy is useful only when the time for instructional delivery or for studying is constrained for all students. Such constraints have nothing to do with the process of learning. To adopt the best teaching strategy should mean to respect the right of all students to be given the opportunity to achieve mastery, where *opportunity* is redefined to mean sustained exposure to the proper conditions of learning, to include being taught learning strategies (e.g., Namlu, 2003), until achievement has been attained at the level of the individual student. No two students in a classroom will show identical readiness for learning new material. Each student will begin a learning experience with a different history or baseline repertoire. All students deserve to experience a process of learning, however different that process may be manifested across students, that takes each of them to the identical level of achievement.

## REFERENCES


Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change.

*Psychological Review*, *84*, 191-215.

Barnett, S.M., & Ceci, S.J. (2002). When and where do we apply what we learn? A

taxonomy for far transfer. *Psychological Bulletin*, *128*, 612-637.

Cuevas, H.M., Fiore, S.M., Bowers, C.A., & Salas, E. (in press). Fostering constructive

cognitive and metacognitive activity in computer-based complex task training environments. *Computers in Human Behavior*.

Emurian, H.H., & Durham, A.G. (2001). A personalized system of instruction for

teaching Java. In M. Khosrowpour (Ed.), *Managing Information Technology in a Global Economy* (pp. 155-160). Hershey: Idea Group Publishing.

Emurian, H.H., & Durham, A.G. (2002). Enhanced learning on a programmed instruction

tutoring system for JAVA. In M. Khosrowpour (Ed.), *Issues and Trends of IT Management in Contemporary Organizations* (pp. 205-208). Hershey: Idea Group Publishing.

Emurian, H.H., & Durham, A.G. (2003). Computer-based tutoring systems: a behavioral

approach. In J.A. Jacko & A. Sears (Eds.), *Handbook of Human-Computer Interaction* (pp. 677-697). Mahwah, NJ: Lawrence Erlbaum & Associates.

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.

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.

Greer, R.D., & McDonough, S.H. (1999). Is the learn unit a fundamental measure of

pedagogy? *The Behavior Analyst*, *22*, 5-16.

Greer, R.D. (2002). *Designing Teaching Strategies: An Applied Behavior Analysis*

*Systems Approach*. NY: Academic Press.

Hayes, S.C. (1989). *Rule-Governed Behavior: Cognition, Contingencies, and*

*Instructional Control*. New York: Plenum Press.

Kudadjie-Gyamfi, E., & Rachlin, H. (2002). Rule-governed versus contingency-governed

behavior in a self-control task: effects of changes in contingencies. *Behavioral Processes*, *57*(1), 29-35.

Mayer, R.E. (2002). *The Promise of Educational Psychology. Volume II. Teaching for*

*Meaningful Learning*. Upper Saddle River, NJ: Pearson Education, Inc.

Namlu, A.G. (2003). The effect of learning strategy on computer anxiety. *Computers in*

*Human Behavior*, *19*, 565-578.

Novick, L.R. (1990). Representational transfer in problem solving. *Psychological*

*Science*, *1*, 128-132.

Potosky, D. (2002). A field study of computer efficacy beliefs as an outcome of training:

the role of computer playfulness, computer knowledge, and performance during training. *Computers in Human Behavior*, *18*, 214-255.

Sackett, P.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.

Sternberg, R.J., & Lyon, G.R. (2002). Making a difference to education: will psychology

pass up the chance? *Monitor on Psychology*, *33*, retrieved on 9/28/2003. URL: http://www.apa.org/monitor/julaug02/difference.html

Torkzadeh, G., & Van Dyke, T.P. (2002). Effects of training on Internet self-efficacy and

computer user attitudes. *Computers in Human Behavior*, *18*, 479-494.

Veenman, M.V.J., Prins, F.J., & Elshout, J.J. (2002). Initial inductive learning in a

complex computer simulated environment: the role of metacognitive skills and intellectual ability. *Computers in Human Behavior*, *18*, 327-241.

Young, J.D. (1996). The effect of self-regulated learning strategies on performance in

learner controlled computer-based instruction. *Educational Technology Research and Development*, *44*, 17-27.

Zimmerman, B.J. (1994). Dimensions of academic self-regulation: a conceptual

framework for education. In D.H. Schunk and B.J. Zimmerman (Eds.), *Self-Regulation of Learning and Performance* (pp. 3-21). Hillsdale, NJ: Erlbaum.


## ENDNOTES

[1] http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/TutorLinks.html.