

Computers in Human Behavior, accepted

**Managing Programmed Instruction and Collaborative Peer Tutoring in the Classroom:
Applications in Teaching Java™**

Henry H. Emurian, Heather K. Holden, & Rachel A. Abarbanel

Information Systems Department
UMBC
1000 Hilltop Circle
Baltimore, Maryland 21250

Voice: 410-455-3206

Fax: 410-455-1073

Email: emurian@umbc.edu

*Notice: This is the authors' version of a work that was accepted for publication in *Computers in Human Behavior*. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication.*

Abstract

To fulfill part of the course requirements, 34 undergraduate students in two courses completed an online programmed instruction tutor as the first technical training exercise in a Java™ programming course designed for information systems majors. The tutor taught a simple JApplet program to display a text string within a browser window on the Web. Students in the first course next participated in a collaborative peer tutoring session, based on the JApplet program, followed by a lecture on the program and by successfully running the program on the Web. For the second course, the peer tutoring session was scheduled after the lecture and after successfully running the program. Students in both courses completed tests of far transfer (“meaningful learning”) and software self-efficacy before using the tutor and on several subsequent occasions following that initial learning. Students in the second course also completed a 4-item scale to assess the development of classification and functionality knowledge regarding elements of the program. Students in both courses showed progressive improvement in all performance measures across the several assessment occasions. Students’ positive ratings of the effectiveness of both the tutor and the collaborative peer tutoring supported the value of these learning experiences in a technical knowledge domain. The results of this study, based on student performance observed within the context of the classroom, show the importance of providing a range of synergistic learning experiences that culminate in a level of skill and confidence that prepares and motivates all students for advanced instruction in Java. They also show how to manage the instructional techniques in the classroom to accomplish that educational outcome.

Keywords: Programmed instruction, collaborative peer tutoring, interteaching, Java training.

INTRODUCTION

Kelleher and Pausch (2005) begin their presentation of a taxonomy of tools and techniques intended to lower the barriers to learning computer programming with the following observation: “Learning to program can be very difficult for beginners at all ages” (p. 83). This is a conclusion shared by reviewers of the instructional literature in programming (Robins, Rountree, & Rountree, 2003) and by others concerned with teaching computer programming (e.g., Traynor & Gibson, 2004). Additionally, it has been acknowledged that the complexity and instability of Java, in particular, pose unique challenges to educators and students alike (Roberts, 2004). Although skill in computer programming is acknowledged to be valuable for information science students (Forgionne, 1991), educators in the discipline recognize that students may sometimes select management information systems (MIS) and related academic majors to avoid the programming demands of a computer science curriculum (Gill & Holton, 2006). Moreover, the diversity of a typical freshman class in computer programming is highlighted by Koen (2005): “Freshman are very diverse with respect to their entering computer skills – some are state computer champions, while others have never touched a computer before” (p. 599). In response to these challenges, the research reported here is an attempt to improve Java instruction for novice college students, as evidenced by enhanced student knowledge and programming confidence, by combining students’ use of an individualized tutoring system with collaborative peer learning as the initial components of a computer programming course.

The literature regarding the content and pedagogy of introductory computer programming is extensive. It includes consideration of logical constructions and flow of control (Papert, 1980), intelligent computer assisted instruction (Anderson & Skwarecki, 1986), approaches to classroom teaching and student learning (Mayer, 1988), emphasis on mathematics and

algorithms (Hu, 2006), and supportive programming environments such as BlueJ (Kolling, Quig, Patterson, & Rosenberg, 2003), DrJava (Hsia, Simpson, Smith, & Cartwright, 2005), Problem-Based Learning (Tsang & Chan, 2004), and the Environment for Learning to Program (Truong, Bancroft, & Roe, 2005), among others. Although complementary to it, the present work falls outside the scope of that important stream of research by focusing on instructional tactics that are intended to promote mastery of one particular computer program at the level of the individual student. This is accomplished by assuming that the student has no prior experience with the knowledge domain and by hypothesizing that repeated exposure to the identical learning objective with different instructional media will have cumulative beneficial effects on a student's skill and confidence. This is consistent with several recommendations of learning principles to promote retention and transfer of knowledge, which include repeated practice with different instructional modalities (Halpern & Hakel, 2003) and with socially supported interactions (Fox & Hackerman, 2003). In the present study, then, repeated practice is accomplished by the students' exposure to an automated tutoring system, a lecture while writing a program, and collaborative peer tutoring.

Programmed instruction, implemented as an automated tutoring system, has been demonstrated to be an effective learning tool for acquiring initial skill and confidence in Java (Emurian, 2004). With respect to teaching computer programming to college-level students, having students work together in pairs may also facilitate each learner's success in completing introductory computer programming projects using Java (Williams, Wiebe, Yang, Ferzli, & Miller, 2002). Such collaborative pair programming is a type of collaborative learning, which is a social situation in which students teach, coach, and/or evaluate each other within groups of two or more students. A broader collaborative approach to learning Java is suggested by Beck,

Chizhik, and McElroy (2005) who designed instructional exercises that incrementally transitioned introductory programming students from four-member learning cooperators to individual problem solvers. The benefits of collaborative learning when applied to computer programming have also been shown in college sophomores' learning of recursive programming using LISP-LOGO (Jehng, 1997). With the exception of the within-class study by Emurian (2004), the above studies involving collaborative learning show how between-subjects experimental designs may be implemented in the classroom, actual or virtual, to identify teaching approaches that make learning computer programming a less formidable task for introductory college-level students.

Randomized field or controlled trials are considered to be the *gold standard* in educational research (Towne & Hilton, 2004; U.S. Department of Education, 2003). However, there is growing recognition that such trials undertaken within an actual classroom setting are vulnerable to multiple sources of confounding variables that make causal attribution problematic (Brown, 1992; Collins, 1992; Edelson, 2002). Indeed, Williams et al. (2002) reported that the behavior of students in the weekly 3-hour *pair programming* laboratory departed across instructors and across students from what that treatment was intended to be. In acknowledging a bias favoring a disproportionate number of white-male students in a control group, Beck et al. (2005) concluded the following: "...we recognize that it is very difficult to perform the perfect experiment when doing educational research" (p. 474). A similar conclusion was reached by Maki (2004) with respect to evaluating instructional effectiveness for undergraduates in science, technology, engineering, and mathematics (STEM) courses: "True experimental designs are difficult and may be impossible to implement in most educational settings..." (p. 31). Additionally, Saville, Zinn, Neef, Norman, and Ferreri (in press) concluded, in a classroom

within-subjects evaluation of the relative effectiveness of lectures and peer collaboration, that “... it is possible that the present studies contained one or more confounding variables and, consequently, low construct validity” (p. 20). Although there are obviously instances of exemplary experiments (e.g., Jehng, 1997) in this area of research, the challenges often associated with implementing and interpreting the outcomes of investigations conducted with students in the classroom suggest the need to consider alternative methods to generate evidence of instructional effectiveness regarding approaches to study learning in context. A promising alternative has been labeled *design-based research* (Sandoval & Bell, 2004).

Design-based research, which is a type of formative evaluation (Collins, Joseph, & Bielaczyc, 2004), is emerging as an alternative methodology in support of developing and assessing systematic improvements in instructional design within the context of the classroom (Bell, Hoadley, & Linn, 2004; Design-Based Research Collective, 2003). Rather than focusing on effect size differences among alternative instructional interventions, which is the outcome of traditional experimental designs (Mayer, 2004a), design-based research starts with an arguably meritorious educational tactic and attempts to improve upon it, as evidenced by enhanced student performance, over successive offerings of a course. Our most recent research provides a detailed justification of this methodology as applied to introductory students’ learning of Java when supported by a programmed instruction tutoring system (Emurian, 2006a) and by a tutoring system session potentiated by a collaborative peer tutoring interaction (Emurian, 2006b). The latter research has the objective of having all students reach a common competency criterion in Java, rather than identifying a treatment associated with a comparatively improved average student performance, an approach that may not always best model a change process at the level of the individual student (e.g., Hamaker, Dolan, & Molenaar, 2005).

The present study builds upon Emurian (2006b), using information systems students in two undergraduate courses and using an updated version of the Java programmed instruction tutoring system. Students in both courses also participated in a collaborative peer tutoring session, which required the knowledge presented in the tutor and which was based upon the dyadic *interteaching* format suggested by Boyce and Hineline (2002). Software self-efficacy and Java knowledge were assessed prior to using the Java tutor and on several occasions thereafter. The assessment of learning consisted of questions reflecting meaningful learning (Mayer, 2002), which intends to show that students can apply or *transfer* their knowledge to solve new problems (Mayer, 2004b). The rationale for adopting this approach to measuring student performance in the present area of work is presented in Emurian (2005). The number of rule-based questions was also increased in the present study, and a scale of concept formation was used in the second course to determine the students' ability to detect classification and functional similarities and differences between pairs of Java identifiers, keywords, methods, classes, operators, and separators. These designed-based replications are intended to show the generality of the instructional interventions with new students and to show the reliability of the outcomes. They also are intended to show how these techniques may be managed in the college classroom to improve student performance and to yield evidence of effectiveness within the context of *design-based research* that motivates and sustains innovation in education (Bell, 2004).

The studies to be presented are a reflection of the compromises that were made to enable this research to be undertaken within the context of an actual classroom. How the resulting observations of student performance contribute to evidence of instructional effectiveness regarding the learning approaches under consideration will be discussed.

METHOD

Subjects

Undergraduate students in two elective courses in Java participated in this research, which was determined to be exempt from informed consent requirements. Both courses required one prior computer programming course as a prerequisite, and they were intended to be taken by majors in information systems. The first course (IS 247J), entitled *Introduction to Programming Using Java*, was offered during the Spring 2005 semester. The course met for 2.5 hours weekly over 14 weeks. The second course (IS 413), entitled *Graphical User Interface Systems Using Java*, was offered during the Fall 2005 semester. The course met for 75 minutes twice weekly over 14 weeks. Although IS 413 emphasized more graphical user interface objects than did IS 247J, both courses were based on the JApplet class so that students could run their project assignments on the Web.

Prior to working on the Java tutoring system, each student completed a background questionnaire¹ giving gender, age, and number of prior computer programming courses taken. Students also rated their experience in computer programming and Java programming on a ten-point Likert-type scale where 1 = *No experience. I am a novice.* to 10 = *Extensive experience. I am an expert.* For the Spring 2005 course, there were 16 males and 6 female students; for the Fall 2005 course, there were 9 males and 3 females ($\chi^2 = 0.03$, $p > .10$). The reported ages were as follows for the Spring 2005 students (median = 23 years, range 19-40) and the Fall 2005

¹ The questionnaires, the tutor study manual, and the tutor code used in this study are accessible in the following file: <http://nasa1.ifsm.umbc.edu/learnJava/research/2005/2005.html>

students (median = 23 years, range 19-28). A comparison² in students' age between the two courses was not significant ($W = 0.306$, $p = 0.584$). Figure 1 presents, for students in both

Insert Figure 1 About Here

courses, boxplots of the total prior programming courses taken and ratings of experience for computer programming and for Java. The minimum value for the y dimension ("Magnitude") is one because one prior programming course was the minimum prerequisite for students in both courses. Tests were not significant for prior programming courses ($W = 0.911$, $p = 0.350$) and programming experience ($W = 0.268$, $p = 0.612$) and marginally significant for Java experience ($W = 3.869$, $p = .061$).

Materials

Descriptions of the Java tutoring system and its component stages, to include the rationale for its design and use, have appeared in our previous reports (Emurian, 2005, 2006a,b). The current version, however, is based on Java Swing³. The tutor teaches how to write and understand a JApplet program (an "applet") that presents a JLabel object in a browser window on the Web. The Java program consists of 34 items or elements (i.e., keywords, identifiers, methods, classes, separators, etc.) of code that are arbitrarily organized into ten lines for the learner. The program is presented in Appendix A.2, which will be explained in a later section.

² Unless noted otherwise, the Welch (W) robust alternative to the F test was used (Maxwell & Delaney, 2004, p. 134), and the Friedman test was used for repeated measures (Conover, 1971, p. 264). The Kruskal-Wallis test was used for tests of linear trend, which were undertaken by comparing all students' regression slopes with a population of zeros (Maxwell & Delaney, 2004, p. 650). Planned pairwise comparisons reported to be significant were Bonferroni adjusted to control the experimentwise error rate for $p = .05$. The computations were undertaken with SPSS.

³ The tutoring system is freely accessible on the Web:

<http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/TutorLinks.html>.

In brief, the Web-based Java tutor consists of the following stages: (1) introduction and example of the program running in a browser, (2) learning to type an item of code, (3) learning to recognize an item in a list, (4) learning the semantics of an item, (5) learning to type a line of code, (6) learning to recognize a line in a list, (7) learning the semantics of a line, and (8) writing the ten lines of code from memory. The multiple-choice tests for items and lines of code, which are embedded in the tutor, had five answer choices. For an incorrect items test answer by the Fall 2005 students, there was a 5-sec delay or “time-out” in the tutor’s interaction with the learner. For a correct items test answer by the Fall 2005 students, a confirmation window appeared stating a general rule associated with the correct answer or an elaboration of the explanation of the meaning of the item. For both courses, the lines interface had no delay interval or confirmation window.

Figure 2 presents the applet as it appears running in a browser on the Web. This

Insert Figure 2 About Here

is displayed in stage 1 of the tutor.

Figure 3 presents the first interface window for the second item to be learned in stage 4.

Insert Figure 3 About Here

The first item (*import*) has already been learned. When the user selects the enabled *Show Java* button, the second item (*javax.swing.JApplet*) appears in blue. Figure 4 presents the

Insert Figure 4 About Here

explanation display window for the second item. When the user selects the enabled *Test* button, the window in Figure 5 is displayed. There are five answers, and the user must scroll as needed

Insert Figure 5 About Here

to find and select the correct answer. Figure 6 shows the consequence of an incorrect selection

Insert Figure 6 About Here

(top) and a correct selection (bottom). As indicated earlier, if the selection is incorrect, the program pauses for 5 seconds, after which the user may select the *Ok* button to proceed. Notice that the input field at the bottom of the window is disabled. An incorrect selection re-enables the *Show Explanation* button, and the user remains in that cycle until a correct choice has been made. If the selection is correct, a confirmation window appears with additional information to support the selection. The location of the item to be entered is highlighted with x's, and the input field is enabled. After the user selects the *Ok* button, input may be entered. If the input is incorrect, a window appears displaying the correct input, and the user tries again, continuing that cycle until the input is correct. If the input is correct, the item appears in place of the x's, and the *Show Java* button is enabled to begin the learning of the next item.

Figure 7 shows correct user input for the second line in stage 6 of the tutor. Stage 6 is

Insert Figure 7 About Here

functionally similar to the stage 4 item interface, but the level of learning is a line of code.

Figure 8 shows an example window for stage 8, the final stage in the tutor. In the example, the input was incorrect when the user selected *Submit*, and the information window

Insert Figure 8 About Here

was displayed, showing the correct program for the user to review. When the user closes that information window, the input window is cleared for new input. The user stays within that cycle until the program is entered correctly.

The course syllabus informed the students in both courses about the potential content of the first quiz. For IS 247J, the syllabus indicated that the first quiz would duplicate the items, lines, and rules questions that appeared on the pre-tutor and post-learning questionnaires. The items and lines questions came directly from the Java tutor. For IS 413, the syllabus indicated that questions from the Java tutor and the rules questions were only eligible for inclusion on the first quiz. The syllabi also informed the students about the requirement and rationale for using the Java tutor and completing the several assessments prior to the first quiz. Questions that were not eligible to appear on the quiz, such as software self-efficacy and programming experience, were clearly labeled as such.

Appendix A (A.1 and A.2) presents the first *inter-teaching report* for the Fall 2005 students. The report shows the instructions, the Java program, and a set of additional questions

for the students to discuss. The 14 rules questions, explained below, were also presented in this document for the Fall 2005 students to discuss. For the Spring 2005 students, the interteaching report contained only the Java program for discussion. Appendix B presents the 12 rules questions for the Spring 2005 students; there were four answer choices for each question. Appendix C.1 presents the 14 rules questions for the Fall 2005 students; there were five answer choices for each question. For each rule question, the student rated his or her confidence that the selected answer was correct, where $1 = \text{No confidence}$ to $10 = \text{Total confidence}$. Appendix C.2 presents the four pairs of symbols that were rated for classification and functionality; the rating scale anchors are also presented there.

Procedure

The instructional events that were delivered to the students in the present study were similar to those investigated in our previous stream of evaluations, with particular reference to the adoption of interteaching as explained in Emurian (2006b). The innovations in the present work included (1) a programmed instruction tutoring system updated to teach a Java Swing applet, (2) an expanded set of rules test questions to assess meaningful learning, and (3) a scale to rate classification and functionality similarity for four pairs of Java symbols presented to the Fall 2005 students.

Table 1 presents the sequence of events for the Spring 2005 students, and Table 2

Insert Tables 1 and 2 About Here

presents the sequence of events for the Fall 2005 students. The design of each course reflected a compromise that took into consideration the length of a class, the number of class meetings per week, and the assessments that were administered to the students.

At the first class for the Spring 2005 students, the tutor was completed in class after the students completed the pre-tutor questionnaires, which consisted of background information, software self-efficacy ratings, and test questions for items, lines, and rules. At the end of the first class, the interteaching report and the tutor study manual were made available on Blackboard for the students to access any time thereafter. The tutor study manual duplicated the explanations of the items and lines of code that were presented in the tutor itself. At the second class meeting, students were paired for a 45-minute interteaching session, which concluded with each student's posting of an interteaching report on Blackboard. For this first interteaching session, the first author used the course list and created pairs by taking names on the list in the order of presentation. This approach was also used for the Fall 2005 students. All subsequent interteaching sessions in both courses allowed the students to generate the pairs, with the stipulation that partners were not to be repeated.

The post-learning questionnaires were then administered, and that was followed by a lecture on the code⁴. During the lecture, which duplicated the code presented in the tutor and in the interteaching report, the students wrote the program using a Unix text editor. When the Java program had been written and compiled, the HTML file was taught, and that was followed by having the students run their programs in a browser on the Web. Individual support was given so that all students successfully ran the program. At the beginning of the third class, the

⁴ The first author was the instructor for both courses.

questionnaires were again administered, and the questions for items, lines, and rules were graded as a quiz.

For students in the Fall 2005 course, the interteaching session was administered after the lecture. This adjustment to the sequence of events was undertaken as an informal assessment of the generality of the techniques as they might be used in the classroom, rather than as an experiment to determine effect size differences as a function of different event sequences. Additionally, the items and lines questions were not administered as part of the questionnaires, although it was stated in the syllabus that those questions, along with the rules questions, were eligible for inclusion on the first quiz in the course. It was also required that students complete the tutor and the post-tutor and post-lecture questionnaires as homework. The tutor study manual was not made available for this course, but the identical information was available in the online tutor, which was continuously available for both the Spring 2005 and Fall 2005 students. This design allowed for four separate assessments, and the first quiz was not part of those occasions for observation of student performance.

The different sequences and instructions between the Spring and Fall courses also reflected the level of each course in the curriculum, the class time available, and the assessments under consideration. The first course, IS 247J, is a low-level programming course that students typically take as freshmen or sophomores. To enhance learning in that course, it was decided to have the first graded quiz consist only of the items and lines questions that were embedded in the tutor, together with the set of rules test questions. The items and lines questions were administered on all three assessment occasions, and they were also embedded in the Java tutor, where correct responding was required to progress from one instructional unit to another. Students were free to ask questions about the rules test questions as part of the interteaching

session, but the correct answers were not made available. The syllabus clearly stated that the first quiz would consist only of the questions presented in the items, lines, and rules tests.

The second course, IS 413, is a high-level programming course that students typically take as juniors or seniors. Although the syllabus indicated that the items, lines, and rules test questions were eligible for inclusion on the first quiz, that first graded assessment also covered additional introductory material that was presented on the interteaching report. For that reason, not all items, lines, and rules test questions appeared on the first quiz for IS 413. It was also decided that the advanced standing of these students would enable them to complete the Java tutor as homework and to undertake at least some of the assessments as homework to be posted on Blackboard. The fact that this course met for 75 minutes also influenced the design.

RESULTS

To assess the magnitude of the changes over successive occasions, a difference score, D_{ij} , was computed for each student ($D_i = 1, n$) for the two sets of differences for the Spring 2005 course ($D_j = 1, 2$) obtained over the three successive assessment occasions. There were three sets of differences ($D_j = 1, 2, 3$) for the Fall 2005 students over four assessment occasions.

Rules Test Performance

Figure 9 presents total correct rule test answers across the three assessment

Insert Figure 9 About Here

occasions for students in the Spring 2005 course and across the four assessment occasions for students in the Fall 2005 course. The data are sorted in ascending order beginning with the pre-tutor outcomes. The student number identifier on the sorted data is retained for comparison on a

subsequent figure. The bar connects the pre-tutor total with the final total, which was the graded quiz for the Spring 2005 students and the interteaching for the Fall 2005 students. The magnitude of the bar, then, is a visual representation of the magnitude of the change for each student across the respective assessment occasions. Since the focus of this research is on individual student achievement, it was decided to display individual outcomes for the rules and software self-efficacy data. Measures of central tendency may obscure the potential orderliness of changes exhibited by individual students who bring different background histories to the classroom.

For the Spring 2005 course, a Friedman test was significant ($\chi^2 = 34.929$, $p = 0.000$), and a comparison of Di1 and Di2 was significant ($W = 31.751$, $p = 0.000$). A test of linear trend was significant ($\chi^2 = 34.396$, $p = 0.000$). For the Fall 2005 course, a Friedman test was significant ($\chi^2 = 32.491$, $p = 0.000$). Comparisons were significant for Di1 and Di2 ($W = 81.714$, $p = 0.000$), Di2 and Di3 ($W = 41.086$, $p = 0.000$), but not significant for Di1 and Di3 ($W = 0.825$, $p = 0.375$). A test of linear trend was significant ($\chi^2 = 19.803$, $p = 0.000$).

Figure 10 presents boxplots of confidence ratings in the accuracy of a rules test answer,

Insert Figure 10 About Here

for answers that were correct (Right) and incorrect (Wrong) across the three assessment occasions for the Spring 2005 students. The boxplots were constructed from the median rating for correct and incorrect answers for each student; accordingly, the figure is a boxplot of medians for all students across the assessment occasions. One of the 22 students did not make ratings of answers, leaving 21 potential ratings. The number of ratings differs across answer accuracy and assessment occasion because some students did not make one or more correct answers or

incorrect answers on a given occasion. For the Spring 2005 students, the W test was significant for Right answers ($W = 32.602$, $p = 0.000$) and for Wrong answers ($W = 25.721$, $p = 0.000$). For Right answers, pairwise comparisons were significant for all three pairs: pre-tutor and post-learning ($W = 28.705$, $p = 0.000$), post-learning and graded quiz ($W = 15.294$, $p = 0.000$), and pre-tutor and graded quiz ($W = 56.314$, $p = 0.000$). For Wrong answers, pairwise comparisons were significant for all three pairs: pre-tutor and post-learning ($W = 20.516$, $p = 0.000$), post-learning and graded quiz ($W = 8.305$, $p = 0.007$), and pre-tutor and graded quiz ($W = 52.365$, $p = 0.000$). An overall comparison of confidence ratings between Right and Wrong answers was significant ($W = 8.039$, $p = 0.005$).

Figure 11 presents boxplots of confidence ratings in the accuracy of a rules test answer,

Insert Figure 11 About Here

for answers that were correct (Right) and incorrect (Wrong) across the four assessment occasions for the Fall 2005 students. The W test was significant for Right answers ($W = 36.659$, $p = 0.000$) and for Wrong answers ($W = 32.794$, $p = 0.000$). For Right answers, pairwise comparisons were significant for pre-tutor and post-tutor occasions ($W = 37.659$, $p = 0.000$) but not significant for post-tutor and lecture ($W = 0.787$, $p = 0.386$), lecture and interteaching ($W = 5.136$, $p = 0.037^5$), and post-tutor and interteaching ($W = 6.494$, $p = 0.024$) occasions. For Wrong answers, pairwise comparisons were significant for pre-tutor and post-tutor occasions ($W = 74.279$, $p = 0.000$) but not significant for post-tutor and lecture ($W = 0.132$, $p = 0.720$), lecture and interteaching ($W = 0.986$, $p = 0.339$), and post-tutor and interteaching ($W = 1.592$, $p = 0.232$) occasions. An overall

⁵ This is greater than 0.0125, which is the corrected p for four planned pairwise contrasts at the .05 level of significance. This rule applies for all contrasts.

comparison of confidence ratings between Right and Wrong answers was significant ($W = 9.604$, $p = 0.003$).

Software self-efficacy

Figure 12 presents ratings of software self-efficacy for students in the Spring 2005 and

Insert Figure 12 About Here

Fall 2005 courses across the respective assessment occasions for each course. The data within each course were sorted in ascending order across the assessment occasions, and the number of the student in each course corresponds to the number presented in Figure 9.

For the Spring 2005 students, Cronbach's alpha values for each of the three assessment occasions were as follows: pre-tutor ($\alpha = 0.98$, $p < .01$), post-learning ($\alpha = 0.97$, $p < .01$), and graded quiz ($\alpha = 0.96$, $p < .01$). A Friedman test was significant ($\chi^2 = 36.028$, $p = 0.000$). A comparison of Di1 and Di2 was significant ($W = 38.965$, $p = 0.000$). A test of linear trend was significant ($\chi^2 = 30.297$, $p = 0.000$).

For the Fall 2005 students, Cronbach's alpha values for each of the four assessment occasions were as follows: pre-tutor ($\alpha = 0.94$, $p < .01$), post-tutor ($\alpha = 0.96$, $p < .01$), lecture ($\alpha = 0.97$, $p < .01$), and interteaching ($\alpha = 0.97$, $p < .01$). A Friedman test was significant ($\chi^2 = 23.792$, $p = 0.000$). A comparison of Di1 and Di2 was significant ($W = 71.027$, $p = 0.000$), but a comparison of Di2 and Di3 was not significant ($W = 1.329$, $p = 0.264$). A comparison of Di1 and Di3 was significant ($W = 70.451$, $p = 0.000$). A test of linear trend was significant ($\chi^2 = 17.385$, $p = 0.000$).

A comparison of software self-efficacy ratings between the Spring 2005 post-learning occasion and the Fall 2005 interteaching occasion was not significant ($W = 2.002$, $p = 0.168$). A similar comparison between the graded quiz and interteaching occasions was not significant ($W = 2.681$, $p = 0.123$).

Java Scale Ratings: Fall 2005 Students

Figure 13 presents boxplots of classification and functionality ratings for *import* and *new*

Insert Figure 13 About Here

over the four assessment occasions. A Friedman test for classification was not significant ($\chi^2 = 5.660$, $p = 0.129$), but the test for functionality was marginally significant ($\chi^2 = 7.026$, $p = 0.071$). A Friedman test comparison between the classification and functionality ratings was significant ($\chi^2 = 17.043$, $p = 0.000$).

Figure 14 presents boxplots of classification and functionality ratings for *jApplet* and

Insert Figure 14 About Here

JApplet across the four assessment occasions. A Friedman test was significant for classification ($\chi^2 = 10.500$, $p = 0.015$). A comparison of Di1 and Di2 was significant ($W = 9.709$, $p = 0.006$). A comparison of Di2 and Di3 was not significant ($W = 1.398$, $p = 0.250$), and a comparison of Di1 and Di3 was marginally significant ($W = 4.964$, $p = 0.038^6$). A Friedman test was significant for functionality ($\chi^2 = 16.484$, $p = 0.001$). A comparison of Di1 and Di2 was significant ($W =$

⁶ This is greater than 0.0167, which is the corrected p for three planned pairwise contrasts at the .05 level of significance.

10.917, $p = 0.004$). A comparison of Di2 and Di3 was not significant ($W = 2.487$, $p = 0.129$). A comparison of Di1 and Di3 was marginally significant ($W = 4.169$, $p = 0.055$). A Friedman test comparison between the classification and functionality ratings was not significant ($\chi^2 = 0.167$, $p = 0.683$).

Figure 15 presents boxplots of classification and functionality ratings for *start()* and *init()*

Insert Figure 15 About Here

over the four assessment occasions. A Friedman test for classification was not significant ($\chi^2 = 0.494$, $p = 0.920$), and the test for functionality was only marginally significant ($\chi^2 = 6.978$, $p = 0.073$). A Friedman test comparison between the classification and functionality ratings was significant ($\chi^2 = 4.50$, $p = 0.034$).

Figure 16 presents boxplots of classification and functionality ratings for ζ and

Insert Figure 16 About Here

; across the four assessment occasions. A Friedman test was not significant for classification ($\chi^2 = 1.402$, $p = 0.705$), but was significant for functionality ($\chi^2 = 17.468$, $p = 0.001$). A comparison of Di1 and Di2 was not significant ($W = 0.026$, $p = 0.873$). A comparison of Di2 and Di3 was significant ($W = 9.715$, $p = 0.005$). A comparison of Di1 and Di3 was marginally significant ($W = 4.509$, $p = 0.046$). A Friedman test comparison between the classification and functionality ratings was significant ($\chi^2 = 31.410$, $p = .000$).

Items and Lines Tests: Spring 2005 Students

Figure 17 presents boxplots of total errors on the items test and the lines test for students

Insert Figure 17 About Here

in the Spring 2005 course over the three assessment occasions. For the items test errors, a Friedman test was significant ($\chi^2 = 33.643$, $p = .000$). A comparison of Di1 and Di2 was marginally significant ($W = 3.819$, $p = 0.057$). A test for linear trend was significant ($\chi^2 = 34.308$, $p = 0.000$). For the lines test errors, a Friedman test was significant ($\chi^2 = 21.028$, $p = 0.000$). A comparison of Di1 and Di2 was not significant ($W = 0.659$, $p = 0.422$). A test for linear trend was significant ($\chi^2 = 27.362$, $p = 0.000$).

Total Tutor Errors

Nine students within each course completed all tutor stages with interpretable records of errors produced for all stages of the tutor. Failure of the script to write an interpretable data record of errors at the conclusion of a tutor stage was attributable to the student's intentionally or inadvertently stopping the browser processing or using the browser's "Back" button. Figure

Insert Figure 18 About Here

18 presents a scatterplot based upon total tutor errors and total errors on the rules test at the post-learning assessment (Spring 2005) and the post-tutor assessment (Fall 2005). The figure shows the direct relationships graphically, and the correlation was significant for the Fall 2005 students, even for this small sample size. Despite the fact that all students exited the tutor with the

identical level of performance, as programmed by the tutor, transfer of learning was not equivalent. In general, students making few errors during the tutor made comparatively few errors on the subsequent rules test.

Interteaching sessions

Figure 19 presents boxplots of ratings of interteaching effectiveness for the Spring 2005

Insert Figure 19 About Here

students across the five interteaching sessions. Because the number of students attending class differed across the five sessions, the Welch test was used for understanding ($W = 4.441$, $p = 0.004$) and test readiness ($W = 5.581$, $p = 0.001$). A Friedman test comparison of ratings between the two types was significant ($\chi^2 = 29.595$, $p = 0.000$).

Figure 20 presents boxplots of ratings of interteaching effectiveness for the Fall 2005

Insert Figure 20 About Here

students across the four interteaching sessions. For the Fall 2005 students, the Welch test for ratings over the four occasions was not significant for understanding ($W = 0.536$, $p = 0.661$) but was significant for test readiness ($W = 3.794$, $p = 0.019$). A Friedman test comparison of ratings between the two types was significant ($\chi^2 = 10.604$, $p = 0.001$).

Tutor Evaluation

Figure 21 presents, for the Spring 2005 and Fall 2005 students, boxplots of evaluation

Insert Figure 21 About Here

ratings of the tutor. All medians were eight or higher, and there were only three outliers present among the Spring 2005 students. Comparisons between the courses were not significant for ratings of overall impression ($W = 1.329$, $p = 0.259$), effectiveness in learning Java ($W = 2.935$, $p = 0.097$), and usability of the tutor ($W = 0.057$, $p = 0.813$).

DISCUSSION

Students in two Java courses showed gains in rules test performance and software self-efficacy across several learning experiences that began with programmed instruction and that later included collaborative peer tutoring. With very few exceptions, students gave consistently high evaluative ratings regarding the effectiveness and usability of the tutoring system in learning Java and regarding the interteaching sessions in facilitating understanding the Java program and readiness to be tested on the code. These observations extend our previous work (Emurian, 2006b) to an upgraded version of the tutor using Java Swing and to an upgraded and expanded set of rules test questions. They support the generality of the instructional interventions to new groups of students, and thereby provide evidence of process and outcome reliability. The techniques are integrated descendants of programmed instruction (Skinner, 1958; Vargas & Vargas, 1991) and the personalized system of instruction (Keller, 1968; Kulik, Kulik, & Carmichael, 1974), the latter including interactions between students as part of the learning and testing process.

The study shows that interteaching may be effectively managed in the computer programming classroom in different sequential and temporal relationships to lectures, at least,

and perhaps to other instructional events as well. Students accepted the rationale of the various approaches that they were required to engage, irrespective of the ordering, as being intended to promote their skill and confidence early in Java training. The sensitivity of interteaching reports to the difficulty of the material is perhaps evident in the dynamic trend-like decreases observed over the first three sessions for the Spring 2005 students. Taken together, however, the outcomes indicate the value to students of technology education that is focused on the behavior of the learners as they engage in structured rehearsal under a variety of modalities to include programmed instruction, self-regulated learning with a study manual, lectures with hands-on demonstrations, and peer collaboration. As the principal innovation in the present classroom evaluations, the social dimension may have potentiated the students' motivation to learn and may provide the occasion for mutual elaboration of the understanding of simple facts and concepts to exceed what might be accomplished by solitary study (Rittschof & Griffin, 2001; Slavin, 1996). All techniques, moreover, are in furtherance of providing repeated practice in a knowledge domain, and that factor has long been recognized as instrumental to knowledge and skill development and retention (e.g., Salas & Cannon-Bowers, 2001; Swezey & Llaneras, 1997). Indeed, the first of ten principles stated by Halpern and Hakel (2003) to enhance long-term retention and transfer of learning is the following: "The single most important variable in promoting long-term retention and transfer is 'practice at retrieval'" (p. 38). Whether such practice effects are reflected by a power function, however, is controversial (e.g., Heathcote, Brown, & Mewhort, 2000).

Similar to our previous studies (Emurian, 2005, 2006a,b), students' answers on the rules tests improved in accuracy over successive assessments. Since the selection of a correct answer required the application of a general principle, this outcome supports the value of the

instructional approaches to produce *meaningful learning* (Mayer, 2002) or *far transfer of learning* (Barnett & Ceci, 2002). The largest improvement was observed between the pre-tutor occasion and the assessment that occurred next, which came after interteaching for the Spring 2005 students and after using the tutor for the Fall 2005 students. For both courses, however, the information available prior to that next assessment was identical, because the study manual available to support the Spring 2005 interteaching session duplicated the information in the tutor, which was freely and continuously available to all students. That information was textual, and the meaningful learning resulted from a student's engagement with textual information first.

In that latter regard, the design of the tutor text follows many of the guidelines offered by Mayer (2002) to promote *meaningful learning*, which relates to the goal of having students come away from the tutoring experience with knowledge and skill that can be applied to novel situations. Embedded within the tutor are advance organizers; signaling; adjunct questions; immediate feedback for performance accuracy and tested understanding of facts, concepts, and rules; and sequential structure building as a superordinate objective that organizes the learning process within a single conceptual framework: the production and understanding of a Java applet. Many tutor components also explicitly state and amplify generalizable rules, which determine hierarchical class memberships (e.g., *A is a member of B.*) drawing abstracted physical features into relational networks (Hayes, Fox, Gifford, Wilson, Barnes-Holmes, & Healy, 2001, p. 37). A simple example is to teach a learner that class names in Java begin, by convention, with an uppercase letter or that methods written with a return-type term have names that begin with a lowercase letter. These features are included within the context of the student's stepwise, developmental, and active rehearsal throughout her or his interaction with the tutoring system. Although the evidence for and against even the theoretical existence of *far transfer* (i.e., the

ability to solve novel problems) and its mechanism of action is controversial (Barnett & Ceci, 2002), we adopt promising and demonstrably effective instructional design techniques, to include programmed instruction and interteaching, with the goal of helping our students to achieve optimal learning performance.

The knowledge transfer from the tutor to the rules test (i.e., *far transfer*) was incomplete, as evidenced by the significant correlation between total tutor errors and the rules test errors exhibited by the Fall 2005 students on the post-tutor assessment. For the Spring 2005 students, however, the correlation between total tutor errors and the post-learning assessment was not significant, suggesting that the self-study and interteaching had favorable consequences on far transfer performance. This outcome is consistent with the observation that frequent testing and test feedback that is immediate and corrective, such as occurred within the Java tutor itself, may sometimes hinder, rather than help, knowledge acquisition by giving learners an overly optimistic opinion of their competency (Mathan & Koedinger, 2005). Although students may sometimes prefer item-by-item feedback (e.g., Buzhardt & Semb, 2002), it is also acknowledged that multiple-choice recognition tests assess only low-level cognitive processes and may lead to overconfidence in long-term retention (Halpern & Hakel, 2003). Finally, knowledge transfer was also incomplete for the items and lines tests (i.e., *near transfer*) for the Spring 2005 students, and those assessments duplicated the multiple-choice tests that were embedded in the tutor itself.

Similar to observations in Emurian (2006a,b), the students' self-reported confidence in the accuracy of their choices on the rules test questions generally increased over the several successive assessment occasions in both courses. Although the students' informed monitoring of their knowledge development was evident in the higher ratings of confidence in correct answers, in comparison to incorrect answers, the fact that confidence also was shown to increase for

incorrect answers suggests that the ability to differentiate between a correct and incorrect answer was incomplete, if not actually hindered by accumulating experience. As far transfer knowledge improved, the confidence that incorrect answers were, in fact, correct also showed gains. Metacognitive skill, as demonstrated in a *self-reflection phase* of self-regulation (Zimmerman & Tsikalas, 2005), requires accurate monitoring of one's own competency in a knowledge domain, and without feedback for performance accuracy, overconfidence may develop no matter how varied and frequent the learning occasions prior to performance assessment (Locke & Lantham, 2002). Students in science, technology, engineering, and mathematics (STEM) need to know when their understanding is faulty, and they need the instructional experiences that lead to understanding a knowledge domain at a deep level (McCray, DeHaan, & Schuck, 2003).

The present study also introduced a new approach to assessing the students' understanding of properties of Java code. For the Fall 2005 students, four pairs of Java symbols were rated for similarity in classification and functionality as a component of each of the four assessment occasions. The category structures reflect rule-based/category-learning tasks in which the rule to maximize accuracy can be learned through an explicit reasoning process (Maddox, Filoteo, Hejl, & Ing, 2005), similar to the examples given above. The first pair members, *import* and *new*, are classified as keywords in Java, and they have no functional similarity. The second pair members, *jApplet* and *JApplet*, have no classification or functional similarity. The third pair members, *start()* and *init()*, are classified as methods, and they are functionally different. The fourth pair members, *{* and *;*, are classified as separators, and they are functionally different.

The results of the assessments were encouraging as indicators of a dynamic process of the development of relational frames of coordination and hierarchical relations (Hayes, Blackledge, & Barnes-Holmes, 2001) based on the formal properties of the symbols, which come directly

from the design conventions of the Java programming language⁷, and based on the use and consequences of those symbols in a program. Ratings of the second pair (see Figure 14), for example, showed pronounced and accurate drops between the pre-tutor and post-tutor occasions, and they remained low thereafter. Although the spelling of the two symbols is identical, the ordering of upper and lower case letters is inconsistent with Java conventions for classification or functionality. The trend-like rating changes for the first pair (see Figure 13), which show increases in classification and decreases in functionality ratings, suggest the need for an instructional history to promote the accurate understanding of these symbols, and more importantly, the classification and use of symbols yet to be encountered. Although obviously simplistic when compared to such investigations as the learning of abstract coherent and other categories (e.g., Erickson, Chin-Parker, & Ross, 2005), these data do complement and broaden the several assessment approaches used previously in this research, and they further suggest the effectiveness of the several instructional techniques to promote skilled performance that is reflected along several dimensions of learning.

Based upon the social learning theory of Bandura (1977), self-efficacy assessments are now frequently used in the information technology literature to determine a user's confidence in using a variety of computer-related applications (e.g., Emurian, 2004; Hsu & Chiu, 2004; Johnson, 2005). In the present study, software self-efficacy, based upon a student's reported confidence in being able to use the symbols in the program, increased over the several learning and assessment occasions in both courses. Fourteen of the 21 students in the Spring 2005 course showed gains between the post-learning assessment and the graded quiz, showing that self-study contributed to confidence. Taken together with the gains evident over the four assessment

⁷ http://java.sun.com/docs/books/jls/second_edition/html/jTOC.doc.html

occasions for the Fall 2005 students, it can be concluded that students benefit from experience with several instructional tactics, to include programmed instruction, lecture, interteaching, and self-study. Although the predictive influence of self-efficacy on future performance has been questioned (Heggestad & Kanfer, 2005), self-efficacy assessments continue to be viewed as an important indicator of the effectiveness of training programs that are intended to produce both skill and motivation to learn (e.g., Johnson, 2005). The heuristic value of self-efficacy in the present context may be understood in terms of a relational frame of coordination whereby a student can predict the strength of his or her own behavior when presented with an opportunity to use a particular Java symbol in context (Barnes-Holmes, O'Hora, Roche, Hayes, Bissett, & Lyddy, 2001) and when presented with an opportunity for continued skill acquisition.

The instructional approaches investigated here came from several different background streams of techniques and research, to include programmed instruction as discussed in Emurian and Durham (2003) and Emurian, Wang, and Durham (2003), learn unit formulations as discussed in Greer and McDonough (1999) and Greer (2002), and interteaching as discussed in Boyce and Hinline (2002), Saville, Zinn, and Elliott (2005), and Saville et al. (in press). From the perspective of teaching computer programming, these techniques converge on what is increasingly recognized as vital ingredients to facilitate science education, in general (DeHaan, 2005). Among several recommendations of effective learning principles to promote retention and transfer, for example, are repeated practice with different instructional modalities (Halpern & Hakel, 2003) and socially supported interactions (Fox & Hackerman, 2003). Although the discipline of computer programming may differ in important ways from the natural sciences such as physics, biology, and chemistry, the recommended tools of teaching have been demonstrably effective when applied to introductory students' learning of a Java computer program.

The present work falls within the scope of design-based research (Brown, 1992; Design-based Research Collective, 2003). In the present study, for example, the instructional design changed between the Spring 2005 and Fall 2005 courses. In the latter course, the decision was made for the interteaching session to come last in the ordering of instructional events. That decision was influenced by our anecdotal observation and assumption that the most informed questions about the material would arise during interteaching by students who were previously exposed to both the tutoring system and the lecture. Although the randomized trial continues to be advanced as the gold standard by the U.S. Department of Education⁸, there are potential problems in maintaining *treatment integrity* (Lane, Bocian, MacMillan, & Gresham, 2004) in that type of field research, as shown by Williams et al. (2002) and Beck et al. (2005), regardless of how rigorous in intention is the initial methodology. Research in programming pedagogy that follows the gold standard of the randomized field trial (Towne & Hilton, 2004) is sometimes compromised by the difficulty of holding all relevant variables constant across treatment conditions under investigation, especially when the analysis is directed toward student behavior within the context of a classroom (Hoadley, 2004). The process of improving the current instructional systems and techniques over successive semesters, based on the learning performance and usability ratings of students who used the tools in the classroom, is similar to an action research perspective in the field of education, as explained by Elias and Dilworth (2003, p. 296), to systematic replication (Sidman, 1960) and self-corrective methods (Barrett, 2002, p. 68)

⁸ The What Works Clearinghouse was established in 2002 by the U.S. Department of Education's Institute of Education (<http://www.whatworks.ed.gov/>). On the site is posted a set of guidelines for acceptable research, and the randomized control trial is given preference in producing actionable findings for educators: http://www.whatworks.ed.gov/reviewprocess/study_standards_final.pdf. The randomized control trial is also advocated in the guidelines available on the U.S. Department of Education website: <http://www.ed.gov/rschstat/research/pubs/rigorousetid/rigorousetid.pdf>

in behavior analysis, and to a type of formative evaluation (Collins et al., 2004). The merits of this design-based tactic for studying learning in context have been discussed recently in a special issue of *Educational Psychologist* (Sandoval & Bell, 2004), and similar interdisciplinary activities are now evident by collections of papers in recent special issues of *Educational Researcher* (Kelly, 2002) and *The Journal of the Learning Sciences* (Barab & Squire, 2004).

Rather than relying on effect size differences and traditional “horse-race experimental designs” (Mayer, 2004a, p. 16), whose determinations and outcomes are limited by the arbitrary temporal constraints of a treatment condition and which tacitly accept below-average student performance as an outcome, as exemplary for producing evidence of causal influence, *causal wisdom* favors acceptance of collective knowledge originating from ethnographic analyses of student experiences, classroom quasi-experiments, and neuro-cognitive investigations (DeHaan, 2005) to foster a common performance outcome in each and every student. Given the obvious fact that students who are deemed “prepared” for a course show different levels of competency at the start, as evidenced here in the variability shown in the pre-tutor data for rules test performance and software self-efficacy, it is unrealistic to design a course for the optimally prepared student. Providing a range of learning experiences to all students assures, perhaps, that at least some of those experiences will be optimally beneficial for the individual student, thereby promoting a common achievement outcome for each student in a course. A traditional between-subjects randomized experiment, which might evaluate potential effects of different orders of the instructional techniques under investigation here as they might differentially impact average student performance, is considered to be counterproductive within the context of a design-based methodology having the goal of producing optimal achievement for all students when the actual classroom demands flexibility in the administration of various learning experiences with students

exhibiting a wide range of preparatory competencies. The trade-offs in favoring a departure from traditional experimental designs in training evaluation research are cogently presented in Sackett and Mullen (1993).

The instructional techniques investigated here in the classroom are effective for introductory students majoring in information systems. Students value the different modes of learning, and they gain self confidence and skill. As discussed previously, however, research in programmed instruction is scarce (Emurian, 2005). Although questionable pedagogical assumptions that may have impacted instructional design have been advanced to account for the apparent demise of programmed instruction (McDonald, Yanchar, & Osguthorpe, 2005), it is equally plausible that the lack of application of this technique is a reflection of a more general university culture that does not value teaching. As stated by DeHaan (2005), “Indeed, most [faculty members] are likely to face significant disincentives to learn new teaching approaches or to reformulate an introductory course: it requires a large investment of time, and it is a distraction from the focus on research” (p. 264). Advancing a scholarship of teaching within information and communication technology (ICT) disciplines requires consideration of the interactive challenges of the teacher’s motivation and skill to undertake research along with organizational support for such work (Lynch, Sheard, Carbone, & Collins, 2005).

The proposed approach to teaching an applet to introductory students may even seem simplistic to experienced programmers and educators. Although we also have the goal of helping students to learn the syntax and semantics of advanced programming such as recursion, we argue that our approach is deliberately and constructively designed to meet the needs of novice students, those *ineffective novices* who lack experience and self-efficacy in this domain. Even students who used the notable *LISP tutor* (Anderson, Conrad, & Corbett, 1989; Anderson,

Corbett, Koedinger, & Pelletier, 1995) had prior experience with the knowledge domain. Basic knowledge, however, overcomes *tactical deficiencies*, examples of which include erroneous syntax, misspellings, and minor logic errors (Williams & Kessler, 2001, p. 8). Robins et al. (2003) call further attention to this issue due to the fact that no existing studies involved ways to transform an *ineffective novice* into an *effective novice* (p. 165), at least at that time.

A recent exception, however, is the work by Mathan and Koedinger (2005) who discussed the timing of feedback following performance errors in fostering an *intelligent novice*, which is based upon a tutoring model that allows students to make reasonable errors and that teaches error detection and correction skills identified as the outcome of *metacognitive tutoring*. In the present study, no feedback was given for rules test performance, and improvement over successive assessments was a function of the collective experiences of the students. The fact that confidence also increased in incorrect performance has been recognized as a detrimental effect of too many assessments, which may lead to a student's overconfidence in their understanding and retention (Halpern & Hakel, 2003). We also acknowledge the positive impact of automated tutoring systems in other domains, and this is evidenced by the Cognitive Tutors that have been designated as one of five exemplary curricula in K-12 mathematics education by the U.S. Department of Education (Mathan & Koedinger, 2005). General advances in the learning and teaching of school subjects, which are recognized by scholars as one of educational psychology's most productive accomplishments of the past two decades (Mayer, 2004b), also inform this discussion. Finally, complementary approaches to programmed instruction are evident in the development of a web-based Environment for Learning to Program (ELP), which supports novice programmers in their initial skill acquisition in Java and other programming languages (Truong, Bancroft, & Roe, 2005).

Rather than assuming that all students who are enrolled into an introductory programming course should possess desired background competencies, it is suggested here that those skills are often important to be taught. Such skills are necessary to acquire for many students who otherwise may withdraw from programming courses as a result of being overwhelmed and demoralized by the sudden demands to be fluent in symbol manipulation as a prerequisite. Although overlooked, to the best of our knowledge, in the literature of computer programming instruction, cognitive load theorists propose a stepwise instructional design for complex learning where each step is within the student's *proximal zone of development* (van Merriënboer, Kirschner, & Kester, 2003, p. 8). This stepwise approach to mastery in a knowledge domain is also reflected in the *learn unit* formulation of Greer and McDonough (1999). It is likely that those *ineffective novices*, who are exposed to instruction falling outside their proximal zones of development and to overly complex learn units, may drop out of STEM courses. We assume that those learners have no prior familiarity with the knowledge domain, and this assumption contrasts with computer-based tutors applied to teach programming within other contexts (e.g., Anderson et al., 1995). As stated by Woolfolk, Winne, and Perry (2000, p. 384), educators need to impart to students "...a combination of academic learning skills and self-control that makes learning easier, so learners are more motivated; in other words, they have the skill and will to learn" (cited in Martin, 2004).

Although it may well be logically fallacious to affirm a null hypothesis, our work involving Java instruction, at least, is motivated so that all students may achieve a common outcome: no difference among students in achievement. Cognitive models of novice programmers have been characterized as superficial, to include line by line comprehension (Winslow, 1996, p. 18), but it is clear that many students drop out of computer programming

courses long before even that fundamental background is achieved. The characterizing and labeling of failed students and their knowledge structure deficiencies have not been matched by approaches to improving teaching effectiveness in the classroom. Instead, the tendency has been for instructors to blame the students for failing instead of concluding that the pedagogy is faulty (Jenkins, 2001). We aim to observe no difference in individual performance among our students, and such an objective requires a rational pedagogy (Emurian, 2001) that removes arbitrary temporal constraints on learning, such as class time or even a semester's duration. Too much educational research, perhaps, is directed toward optimizing average student performance under these arbitrary and temporal constraints. As suggested in Emurian (2006a), an optimal teaching strategy in the 21st century should be one that respects the right of each and every student to have the opportunity to achieve mastery, where opportunity is taken to mean unlimited exposure to the proper conditions of learning until an achievement outcome has been attained. As stated by Anderson et al. (1995), "It is more meaningful to hold constant the level of mastery required and look at differences in time to achieve that level. This reflects the true gain of an educational technique" (p. 185). We aim for that *true gain* by our students in response to our instructional tactics in technology education.

REFERENCES

- Anderson, J.R., Conrad, F.G., & Corbett, A.T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science*, 13, 467-505.
- Anderson, J.R., Corbett, A.T., Koedinger, K.R., & Pelletier, R. (1995). Cognitive tutors: lessons learned. *Journal of Learning Science*, 4, 167-207.
- Anderson, J.R. and Skwarecki, E. (1986). The automated tutoring of introductory computer programming. *Communications of the ACM*, 29(9), 842-849.
- Bandura, A. (1977). Self-efficacy: toward a unifying theory of behavioral change. *Psychological Review*, 84, 191-215.

- Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground (Introduction to a special issue). *The Journal of the Learning Sciences*, 13(1), 1-14.
- Barnes-Holmes, D., O'Hara, D., Roche, B., Hayes, S.C., Bissett, R.T., & Lyddy, F. (2001). Understanding and verbal regulation (pp. 103-117). In S.C. Hayes, D. Barnes-Holmes, & B. Roche (Eds.). *Relational Frame Theory: A Post-Skinnerian Account of Human Language and Cognition*. New York: Kluwer Academic/Plenum Publishers.
- Barrett, B.H. (2002). *The Technology of Teaching Revisited: A Reader's Companion to B.F. Skinner's Book*. Concord, MA: Cambridge Center for Behavioral Studies.
- 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.
- Beck, L.L., Chizhik, A.W., & McElroy, A.C. (2005). Cooperative learning techniques in CS1: design and experimental evaluation. *Proceedings of the Thirty-Sixth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE 2005, 470-474.
- Bell, P. (2004). On the theoretical breadth of design-based research in education. *Educational Psychologist*, 39(4), 243-253.
- 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.
- Brown, A.L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, 2(2), 141-178.
- Buzhardt, J., & Semb, G.B. (2002). Item-by-item versus end-of-test feedback in a computer-based PSI course. *Journal of Behavioral Education*, 11(2), 89-104.
- Collins, A. (1992). Toward a design science of education. In E. Scanlon & T. O'Shea (Eds.), *New directions in educational technology* (pp. 15-22). New York: Springer-Verlag.
- Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: Theoretical and methodological issues. *The 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.
- Edelson, D.C. (2002). Design research: What we learn when we engage in design. *The Journal of the Learning Sciences*, 11(1), 105-121.
- Elias, M.J., & Dilworth, J.E. (2003). Ecological/developmental theory, context-based best practice, and school-based action research: cornerstones of school psychology training and policy. *Journal of School Psychology*, 41, 293-297.
- Emurian, H.H. (2001). The consequences of e-Learning (Editorial), *Information Resources Management Journal*, April-June, 3-5.
- 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.
- Erickson, J.E., Chin-Parker, S., & Ross, B.H. (2005). Inference and classification learning of abstract coherent categories. *Journal of Experimental Psychology: Learning, Memory, and Cognition*. 31(1), 86-99.
- Forgionne, G.A. (1991). Providing complete and integrated information science education. *Information Processing & Management*, 27(5), 575-590.
- 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.
- Gill, T.G., & Holton, C.F. (2006). A self-paced introductory programming course. *Journal of Information Technology Education*, 5, 95-105.

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

Hamaker, E.L., Dolan, C.V., & Molenaar, P.C.M. (2005). Statistical modeling of the individual: Rationale and application of multivariate stationary time series analysis. *Multivariate Behavioral Research*, 40(2), 207-233.

Hayes, S.C., Blackledge, J.T., & Barnes-Holmes, D. (2001). Language and cognition: constructing an alternative approach within the behavioral tradition. In S.C. Hayes, D. Barnes-Holmes, and B. Roche (Eds.), *Relational Frame Theory: a Post-Skinnerian Account of Human Language and Cognition* (pp. 3-20). New York: Kluwer Academic/Plenum Publishers.

Hayes, S.C., Fox, E., Gifford, E.V., Wilson, K.G., Barnes-Holmes, D., & Healy, O. (2001). Derived relational responding as learned behavior. In S.C. Hayes, D. Barnes-Holmes, and B. Roche (Eds.), *Relational Frame Theory: a Post-Skinnerian Account of Human Language and Cognition* (pp. 21-49). New York: Kluwer Academic/Plenum Publishers.

Heggestad, E.D., & Kanfer, R. (2005). The predictive validity of self-efficacy in training performance: little more than past performance. *Journal of Experimental Psychology: Applied*, 11(2), 84-97.

Heathcote, A., Brown, S., & Mewhort, D.J.K. (2000). The power law repealed: The case for an exponential law of practice. *Psychonomic Bulletin and Review*, 7(2), 187-207.

Hoadley, C.M. (2004). Methodological alignment in design-based research. *Educational Psychologist*, 39(4), 203-212.

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.

Hsu, M.-H., & Chiu, C.-M. (2004). Internet self-efficacy and electronic service acceptance. *Decision Support Systems*, 38(3), 369-381.

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

Jehng, J.-C. J. (1997). The psycho-social processes and cognitive effects of peer-based collaborative interactions with computers. *Journal of Educational Computing Research*, 17(1), 19-46.

- Jenkins, T. (2001). Teaching programming – a journey from teacher to motivator. 2nd Annual LTSN-ICS Conference, London. URL: <http://www.ics.ltsn.ac.uk/pub/conf2001/papers/Jenkins.htm> [Retrieved on 12/6/2005].
- Johnson, R.D. (2005). An empirical investigation of sources of application-specific computer-self-efficacy and mediators of the efficacy-performance relationship. *International Journal of Human-Computer Studies*, 62(6), 737-758.
- Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2), 83-137.
- Keller, F.S. (1968). Goodbye teacher... *Journal of Applied Behavior Analysis*, 1, 79-89.
- Kelly, A.F. (2002). Research as design (Introduction to special issue: The role of design in educational research). *Educational Researcher*, 32(1), 3-4.
- Koen, B.V. (2005). Creating a sense of “presence” in a web-based PSI course: the search for Mark Hopkins’ log in a digital log. *IEEE Transactions on Education*, 48(4), 599-604.
- Kolling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Journal of Computer Science Education*, 14(Dec), 1-12.
- Kulik, J.A., Kulik, C.-L., & Carmichael, K. (1974). The Keller plan in science teaching. *Science*, 183, 379-383.
- Lane, K., Bocian, K.M., MacMillan, D.L., & Gresham, F.M. (2004). Treatment integrity: an essential – but often forgotten – component of school-based interventions. *Preventing School Failure*, 48(3), 36-43.
- Locke, E.A., & Lantham, G.P. (2002). Building a practically useful theory of goal setting and task motivation. *American Psychologist*, 57(9), 705-717.
- Lynch, J., Sheard, J., Carbone, A., & Collins, F. (2005). Individual and organisational factors influencing academics’ decisions to pursue the scholarship of teaching ICT. *Journal of Information Technology Education*, 4, 219-236.
- Maddox, W.T., Filoteo, J.V., Hejl, K.D., & Ing, A.D. (2005). Category number impacts rule-based but not information-integration category learning: further evidence for dissociable category-learning systems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30(1), 227-245.
- Maki, W.S. (2004). Evaluating outcomes. *Invention and Impact: Building Excellence in Undergraduate Science, Technology, Engineering and Mathematics (STEM) Education* (pp. 27 – 32). Workshop report published by AAAS. URL: http://www.aaas.org/publications/books_reports/CCLI/PDFs/02_AER_Maki.pdf [Retrieved on 1/17/06].

- Martin, J. (2004). Self-regulated learning, social cognitive theory, and agency. *Educational Psychologist, 39*(2), 135-145.
- Mathan, S.A., & Koedinger, K.R. (2005). Fostering the intelligent novice: learning from errors with metacognitive tutoring. *Educational Psychologist, 40*(4), 257-265.
- Maxwell, S.E., & Delaney, H.D. (2004). *Designing experiments and analyzing data: a model comparison perspective. Second edition.* Mahwah, NJ: Lawrence Erlbaum Associates.
- Mayer, R.E. (1988). *Teaching and learning computer programming: Multiple research perspectives.* Hillsdale, NJ: Lawrence Erlbaum Associates.
- 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. (2004a). Should there be a three-strikes rule against pure discovery learning? *American Psychologist, 59*(1), 14-19.
- Mayer, R.E. (2004b). Teaching of subject matter. *Annual Review of Psychology, 55*, 715-744.
- McCray, R.A., DeHaan, R.L., & Schuck, J.A. (2003). *Improving undergraduate instruction in science, technology, engineering, and mathematics.* Washington, DC: The National Academies Press.
- McDonald, J.K., Yanchar, S.C., & Osguthorpe, R.T. (2005). Learning from programmed instruction: examining implications for modern instructional technology. *Educational Technology Research & Development, 53*(2), 84-98.
- Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas.* New York: Basic Books.
- Rittschof, K.A., & Griffin, B.W. (2001). Reciprocal peer tutoring: re-examining the value of a cooperative learning technique to college students and instructors. *Educational Psychology, 21*(3), 313-331.
- Roberts, E. (2004). Resources to support the use of Java in introductory computer science. *Proceedings of the 35th SIGCSE technical symposium on computer science education* (pp. 233-234), Norfolk, VA. URL: <http://portal.acm.org/citation.cfm?id=971384> [Retrieved on 11/21/05].
- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172.
- 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.

- Sandoval, W.A., & Bell, P. (2004). Design-based research methods for studying learning in context: introduction. *Educational Psychologist*, 39(4), 199-201.
- Saville, B.K., Zinn, T.E., & Elliott, M.P. (2005). Interteaching versus traditional methods of instruction: A preliminary analysis. *Teaching of Psychology*, 32(3), 161-163.
- Saville, B.K., Zinn, T.E., Neef, N.A., Norman, R.V., & Ferreri, S.J. (in press). A comparison of interteaching and lecture in the college classroom. *Journal of Applied Behavior Analysis*.
- Sidman, M. (1960). *Tactics of Scientific Research*. New York: Basic Books.
- Skinner, B.F. (1958). Teaching machines. *Science*, 128, 969-977.
- Slavin, R.E. (1996). Research on cooperative learning and achievement: what we know, what we need to know. *Contemporary Educational Psychology*, 21, 43-69.
- Swezey, R.W., & Llaneras, R.E. (1997). Models in training and instruction. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (pp. 514-577). New York: Wiley.
- Towne, L., & Hilton, M. (Eds.) (2004). Implementing Randomized Field Trials in Education: Report of a Workshop. Committee on Research in Education. Center for Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.
- Traynor, D., & Gibson, J.P. (2004). Towards the development of a cognitive model of programming: A software engineering approach. *16th meeting of the psychology of programming interest group*. URL: <http://www.cs.may.ie/~dtraynor/papers/PPIGarticle.pdf> [Retrieved on 8/10/2005].
- 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.
- U.S. Department of Education (2003). Identifying and implementing educational practices supported by rigorous evidence. URL: <http://www.excelgov.org/evidence> [Retrieved on 11/21/05].
- van Merriënboer, J.J.G., Kirschner, P.A., & Kester, L. (2003). Taking the load off a learner's mind: instructional design for complex learning. *Educational Psychologist*, 38(1), 5-13.
- Vargas, E.A., & Vargas, J.E. (1991). Programmed instruction: what it is and how to do it. *Journal of Behavioral Education*, 1(2), 235-251.

Williams, L.A., & Kessler, R.R. (2001). Experiment with industry's "pair-programming" model in the computer science classroom. *Computer Science Education*, 11(1), 7-20.

Williams, L.A., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education*, 12(3), 197-212.

Winslow, L.E. (1996). Programming pedagogy – a psychological overview. *SIGCSE Bulletin*, 28, 17-22.

Woolfolk, A.E., Winne, P.H., & Perry, N.E. (2000). *Educational Psychology* (Canadian edition). Scarborough, Ontario, Canada: Allyn and Bacon Canada.

Zimmerman, B.J., & Tsikalas, K.E. (2005). Can computer-based learning environments (CBLEs) be used as self-regulatory tools to enhance learning? *Educational Psychologist*, 40(4), 267-271.

APPENDIX A.1

Copy only this page and paste it in the Discussion Board.

**IS 413
Fall 2005**

Interteaching Report #1

Date:

Your name:

Your partner's name:

How effective was this session in helping you to learn the material?

1 = Not at all effective. The session did not contribute to my learning of the material.

10 = Totally effective. The session contributed to my learning of the material.

(Not effective) **1 2 3 4 5 6 7 8 9 10** (Totally effective)

Enter one number that describes the effectiveness for you: ____.

How confident are you that you could answer all questions correctly if you were tested on this program right now?

1 = Not at all confident. I could not answer any question correctly.

10 = Totally confident. I could answer all the questions correctly.

(Not confident) **1 2 3 4 5 6 7 8 9 10** (Totally confident)

Enter one number that describes your confidence: _____.

If you have questions about this material, write them here:

APPENDIX A.2

Interteaching Objectives for Fall 2005

The below questions may appear on the quiz. The questions embedded in the Java tutor are also eligible to appear on the quiz.

You should understand the components of the below program at a level given in the Java Tutor. Also read the material posted in Unit 1 and Unit 2 (1-4) of the online course material.

Discuss these components with the intention to understand the specific item and any general principle that is reflected in an item or collection of items. An example of a general principle would be to begin the name of a class with a capital letter.

```
import javax.swing.JApplet;
import javax.swing.JLabel;
public class MyProgram extends JApplet {
    JLabel myLabel;
    public void init() {
        myLabel = new JLabel("This is my first program.");
        myLabel.setVisible(true);
        getContentPane().add(myLabel);
    }
}
```

You should be able to answer the following questions:

1. What is a class?
2. What is a statement? Give an example.
3. What is a separator? Give an example.
4. What is a keyword? Give an example.
5. What is an identifier?
6. What does it mean that methods may be inherited from a superclass?
7. What is the meaning of override?
8. How can you identify a series of characters as the name of a method?
9. What is a constructor method? What properties of the syntax make it a constructor method?
10. Describe the position and functions of the terms in a statement that uses a method to change a property of an object.

The rules questions presented in Appendix C.1 also were presented in this interteaching report.

APPENDIX B**Rules Test for Spring 2005**

Circle the best choice that you can make.

1. Which of the following lines most likely would be used to create a shorthand notation for the JFrame class, which is built-in to Java?
 - a. import ../class/JFrame;
 - b. import JFrame.class;
 - c. import java.awt.JFrame.class;
 - d. import javax.swing.JFrame;

2. Which of the following lines most likely would be used to construct an instance of the Button class?
 - a. myButton = new Button("Hello");
 - b. Button = 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. myContainer.Add(myCheckBox);
 - b. container.Add(CheckBox);
 - c. add(myCheckBox);
 - d. add(container.CheckBox);

4. Which of the following lines most likely overrides a method that is contained in the Applet.class file?
 - 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. construct a TextField object, declare a TextField object, add a TextField object to a container.
 - b. declare a TextField object, construct 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.
6. Given the line, **public class MyTextArea extends JTextArea {**, which of the following statements is correct?
- a. JTextArea is a superclass of MyTextArea.
 - b. JTextArea is a subclass of MyTextArea.
 - c. MyTextArea is a superclass of the extends class.
 - d. MyTextArea is a subclass of the JText class.
7. Which one of the below lines declares myJFrame as a potential instance of the JFrame class?
- a. myJFrame extends JFrame.
 - b. JFrame myJFrame:
 - c. JFrame myJFrame;
 - d. myJFrame JFrame;
8. Given the following code: **public class MyJButton extends JButton { ...** which one of the below would be the name of the file that contains this program for compilation?
- a. MyJButton.file
 - b. JButton.java
 - c. JButton.file
 - d. MyJButton.java
9. Which of the following lines would most likely add a JTextField object to a JPanel object?
- a. JPanel.add(JTextField);
 - b. JPanel.add(myJTextField);
 - c. myJPanel.add(JTextField);
 - d. myJPanel2.add(myJTextField2);
10. A Java JApplet program has two methods written in the class. The methods are not nested. What is the total number of braces, { and } added together, that are needed for this program.
- a. 6
 - b. 9
 - c. 3
 - d. 4

11. A programmer intends to construct a JLabel object in a program, but an import statement was not used. Which of the below lines most likely reflects the declaration of the JLabel object?
- a. JLabel myJLabel;
 - b. javax.swing.JLabel myJLabel;
 - c. myJLabel javax.swing.JLabel;
 - d. javax.swing.myJLabel JLabel;
12. Which of the following most likely would be used to change the color of a JApplet container?
- a. setBackground(JApplet = orange);
 - b. this.SetBackground(blue);
 - c. this.setBackground(Color.yellow);
 - d. SetBackground(Container.red);

APPENDIX C.1**Rules Test for Fall 2005**

1. Which of the following lines most likely would be used to create a shorthand notation for the compiler to locate the JFrame class, which is built-in to Java?
 - a. import ../class/JFrame;
 - b. access JFrame.class;
 - c. import javax.swing.JFrame;
 - d. import java.awt.JFrame.class;
 - e. append javax.swing.JFrame;

2. Which of the following lines most likely would be used to construct an instance of the JButton class?
 - a. Button = new JButton("Hello");
 - b. myJButton = new JButton("Click Me");
 - c. Button = new Button("Hello");
 - d. myButton = Button.class("Hello");
 - e. myButton = new JButton("Click Me").

3. Which of the following lines most likely would be used to add a Checkbox object to a content pane?
 - a. getContentPane.Add(myCheckBox);
 - b. container.Add(CheckboxObject);
 - c. add(container.Checkbox);
 - d. getContentPane().add(myBox);
 - e. 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)
 - e. Public void stop() { lines of Java code here }

5. Which of the following sequences is correct?
 - a. declare a JTextField object, construct a JTextField object, add a JTextField object to a container.

- b. construct a JTextField object, declare a JTextField object, add a JTextField object to a container.
 - c. declare a JTextField object, add a JTextField object to a container, construct a JTextField object.
 - d. add a JTextField object to a container, declare a JTextField object, construct a JTextField object.
 - e. add a JTextField object to a container, construct a JTextField object, declare a JTextField object.
6. Given the line, **public class MyTextArea extends JTextArea {**, which of the following statements is correct?
- a. JTextArea is a subclass of MyTextArea.
 - b. MyTextArea is a superclass of the extends class.
 - c. JTextArea is a superclass of MyTextArea.
 - d. MyTextArea is a subclass of the JText class.
 - e. JTextArea is a class of MyTextArea.
7. Which one of the below lines declares myList as a potential instance of the JList class?
- a. myList JList;
 - b. JList myJList;
 - c. JList myList;
 - d. myJList JList;
 - e. JList myList.
8. Given the following code: **public class MyJSlider extends JSlider { ...** which one of the below would be the name of the file that contains this program for compilation?
- a. MyJslider.java
 - b. JSlider.java
 - c. MyJSlider.javax
 - d. myJSlider.java
 - e. MyJSlider.java
9. Which of the following lines would most likely add a JScrollPane object to a JPanel object?
- a. JPanel.add(JScrollPane);
 - b. JPanel.add(myJScrollPane);
 - c. myJPanel.add(JScrollPane);
 - d. JScrollPane.add(JPanelObject);
 - e. myJPanel2.add(myJScrollPane1);

10. A Java JApplet program has two methods written in the class. The methods are not nested. What is the total number of braces, { and } added together, that are needed for this program.
- a. 9
 - b. 6
 - c. 3
 - d. 4
 - e. 2
11. A programmer intends to use the TableColumn class, which is located in the table package. Which statement below is correct to use in the program?
- a. `import javax.swing.table.TableColumn`
 - b. `javax.swing.table.TableColumn;`
 - c. `import javax.swing.TableColumn;`
 - d. `import javax.swing.table.TableColumn;`
 - e. `import TableColumn;`
12. Which of the following most likely would be used to make a JButton object invisible?
- a. `JButton.setBackground(invisible);`
 - b. `myButton.setVisible(opaque);`
 - c. `myButton.setVisible(false);`
 - d. `JButton.SetBackground();`
 - e. `MyButton.setInVisible(true);`
13. Which of the following is a correct class definition?
- a. `public class StandardJFrame extends JApplet }`
 - b. `public StandardJFrame is-a JFrame {`
 - c. `public class MyLabel extends JLabel {`
 - d. `class public MyFrame extends JFrame {`
 - e. `public class StandardJFrame extends JFrame {`
14. Which of the below shows a correct form of a method?
- a. `public void int start() {statements}`
 - b. `public class stop() {statements}`
 - c. `public void Init() {statements}`
 - d. `public void stopRunning() {statements}`
 - e. `public init() destroy() [statements]`

APPENDIX C.2

The below questions are based on the design of the Java programming language and associated conventions of the language. **Functionality** refers to the effects of an item in a Java program. **Classification** refers to: keyword, identifier, method, separator, and operator. Give the most informed rating that you can at this point in your understanding of Java. Give one number for Functionality and one number for Classification.

1. How similar to each other are the following two items in terms of functionality and classification?

(1) import (2) new

Functionality: Not Similar 1 2 3 4 5 6 7 8 9 10 Highly Similar

Classification: Not Similar 1 2 3 4 5 6 7 8 9 10 Highly Similar

2. How similar to each other are the following two items in terms of functionality and classification?

(1) jApplet (2) JApplet

Functionality: Not Similar 1 2 3 4 5 6 7 8 9 10 Highly Similar

Classification: Not Similar 1 2 3 4 5 6 7 8 9 10 Highly Similar

3. How similar to each other are the following two items in terms of functionality and classification?

(1) start() (2) init()

Functionality: Not Similar 1 2 3 4 5 6 7 8 9 10 Highly Similar

Classification: Not Similar 1 2 3 4 5 6 7 8 9 10 Highly Similar

4. How similar to each other are the following two items in terms of functionality and classification?

(1) { (2) ;

Functionality: Not Similar 1 2 3 4 5 6 7 8 9 10 Highly Similar

Classification: Not Similar 1 2 3 4 5 6 7 8 9 10 Highly Similar

FIGURE CAPTIONS

- Figure 1. Boxplots of the total prior programming courses taken and ratings of experience for computer programming and for Java. The circles is an outlier, and the triangles are extreme values.
- Figure 2. The applet as it appears running in a browser window on the Web.
- Figure 3. The first interface window for the second item to be learned in stage 4. Selection of the enabled *Show Java* button displays that second item.
- Figure 4. The explanation display window for the second item to be learned in the program.
- Figure 5. The multiple-choice test window for the second item.
- Figure 6. Consequence of an incorrect selection (top) and a correct selection (bottom) for the multiple-choice test.
- Figure 7. Correct user input for the second line in stage 6 of the tutor. This stage is functionally similar to stage 4, but the level of learning is a line of code. When the user presses the keyboard *Enter* key, the input will replace the x's in the display of the program.
- Figure 8. The input window for stage 8. On the left is user input that is incorrect. When the *Submit* button was selected, the overlay window was displayed, showing the correct code. When the *Noted* button is selected, that window disappears and the input window is cleared.
- Figure 9. Total correct rules test answers across the three assessment occasions for students in the Spring 2005 course and across the four assessment occasions for students in the Fall 2005 course.
- Figure 10. Boxplots of confidence ratings in the accuracy of a rules test answer for answers that were correct (Right) and incorrect (Wrong) across the three assessment occasions for the Spring 2005 students. The circles are outliers.
- Figure 11. Boxplots of confidence ratings in the accuracy of a rules test answer for answers that were correct (Right) and incorrect (Wrong) across the four assessment occasions for the Fall 2005 students. The circle is an outlier.
- Figure 12. Ratings of software self-efficacy for students in the Spring 2005 and Fall 2005 courses across the respective assessment occasions for each course.
- Figure 13. Boxplots of classification and functionality ratings for *import* and *new* over the four assessment occasions for the Fall 2005 students. The circle is an outlier.

- Figure 14. Boxplots of classification and functionality ratings for *jApplet* and *JApplet* over the four assessment occasions for the Fall 2005 students. The circles are outliers.
- Figure 15. Boxplots of classification and functionality ratings for *start()* and *init()* over the four assessment occasions for the Fall 2005 students.
- Figure 16. Boxplots of classification and functionality ratings for *{* and *;* over the four assessment occasions for the Fall 2005 students. The circles are outliers, and the triangles are extreme values.
- Figure 17. Boxplots of total errors on the items test and the lines test over the three assessment occasions for the Spring 2005 students. The circles are outliers, and the triangles are extreme values.
- Figure 18. A scatterplot based upon total tutor errors and total errors on the rules test at the post-learning assessment (Spring 2005) and the post-tutor assessment (Fall 2005).
- Figure 19. Boxplots of ratings of interteaching effectiveness for the Spring 2005 students across the five interteaching sessions. The circles are outliers, and the triangles are extreme values.
- Figure 20. Boxplots of ratings of interteaching effectiveness for the Fall 2005 students across the four interteaching sessions. The circle is an outlier, and the triangles are extreme values.
- Figure 21. Boxplots of evaluation ratings of the tutor for the Spring 2005 and Fall 2005 students. The circles are outliers.

Table 1
Spring 2005
Sequence of Events

IS 247J: Introductory Programming with Java	Spring 2005 (<i>n</i> = 22)
	Sequence of Events Questionnaires: SSE, Items, Lines, and Rules
150-min Classes	
Class 1 2/2/2005	<ol style="list-style-type: none"> 1. Pre-Tutor Questionnaires 2. Tutor 3. Access Interteaching Report from Blackboard
Homework	<ol style="list-style-type: none"> 1. Access to the Tutor Study Manual 2. Optional access to the Tutor
Class 2 2/9/2005	<ol style="list-style-type: none"> 1. Interteaching Session (45 Minutes) 2. Post-Learning Questionnaires 3. Lecture 4. Run the Program
Class 3 2/16/05	<ol style="list-style-type: none"> 1. Final Questionnaires <ul style="list-style-type: none"> ○ Test Credit for Items, Lines, and Rules.

Table 2
Fall 2005
Sequence of Events

IS 413: Graphical User Interface Systems Using Java	Fall 2005 (<i>n</i> = 12)
	Sequence of Events Questionnaires: SSE, Rules, and Java Scale
75-min Classes	
Class 1 8/31/2005	<ol style="list-style-type: none"> 1. Course Orientation 2. Pre-Tutor Questionnaires
Homework	<ol style="list-style-type: none"> 1. Tutor 2. Post-Tutor Questionnaires <ul style="list-style-type: none"> • Submit to Blackboard
Class 2 9/7/2005	<ol style="list-style-type: none"> 1. Lecture 2. Run the Program
Homework	<ol style="list-style-type: none"> 1. Post-Lecture Questionnaires <ul style="list-style-type: none"> • Submit to Blackboard 2. Access Interteaching Report from Blackboard 3. Optional access to the Tutor
Class 3 9/12/2005	<ol style="list-style-type: none"> 1. Interteaching Session: 45 minutes 2. Post-Interteaching Questionnaires

Figure 1.

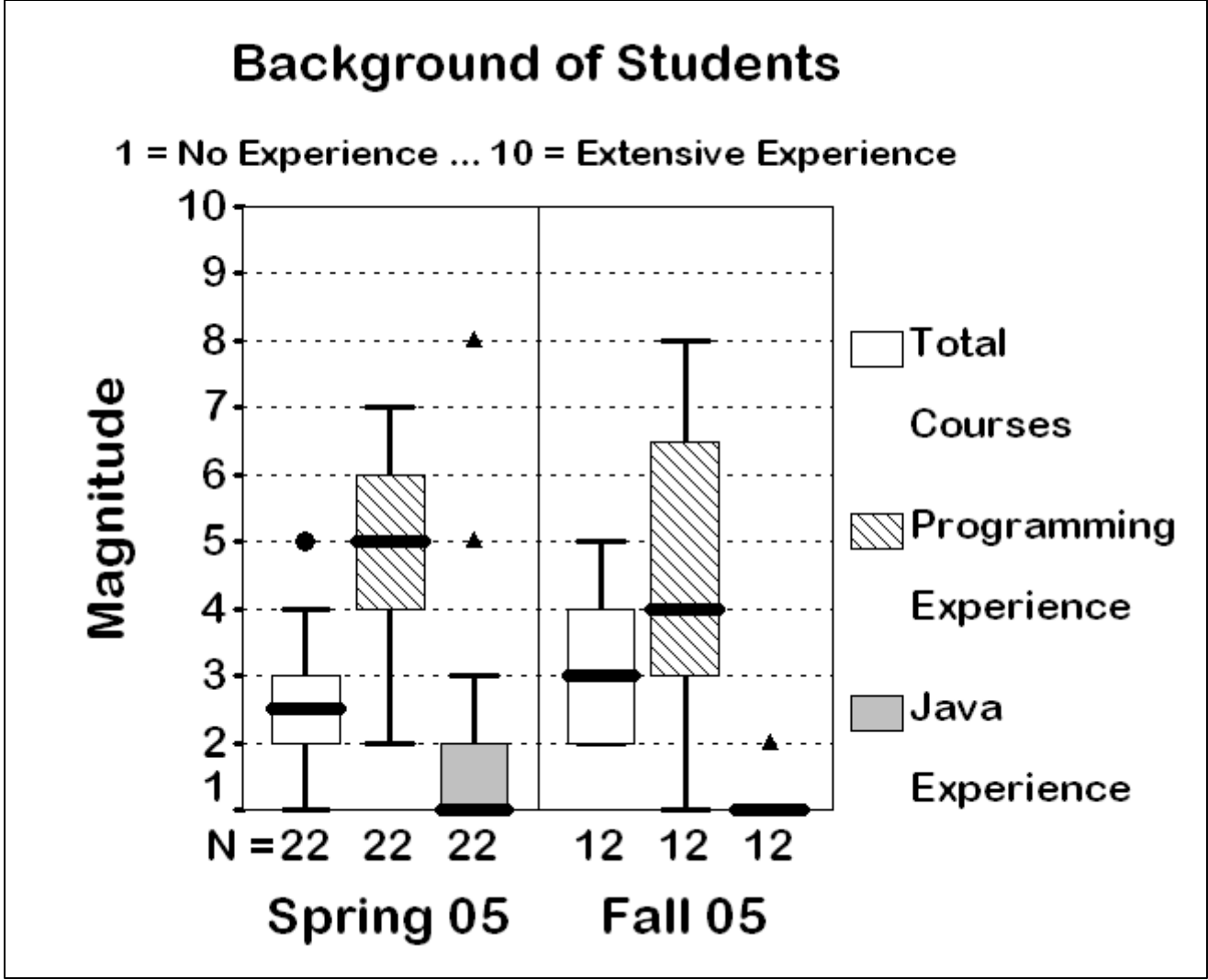


Figure 2.

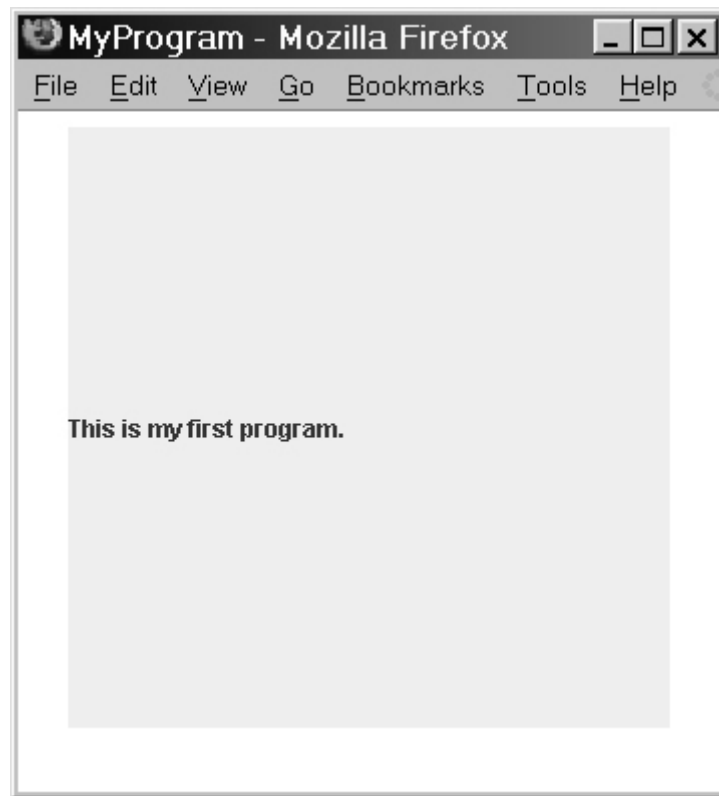


Figure 3.

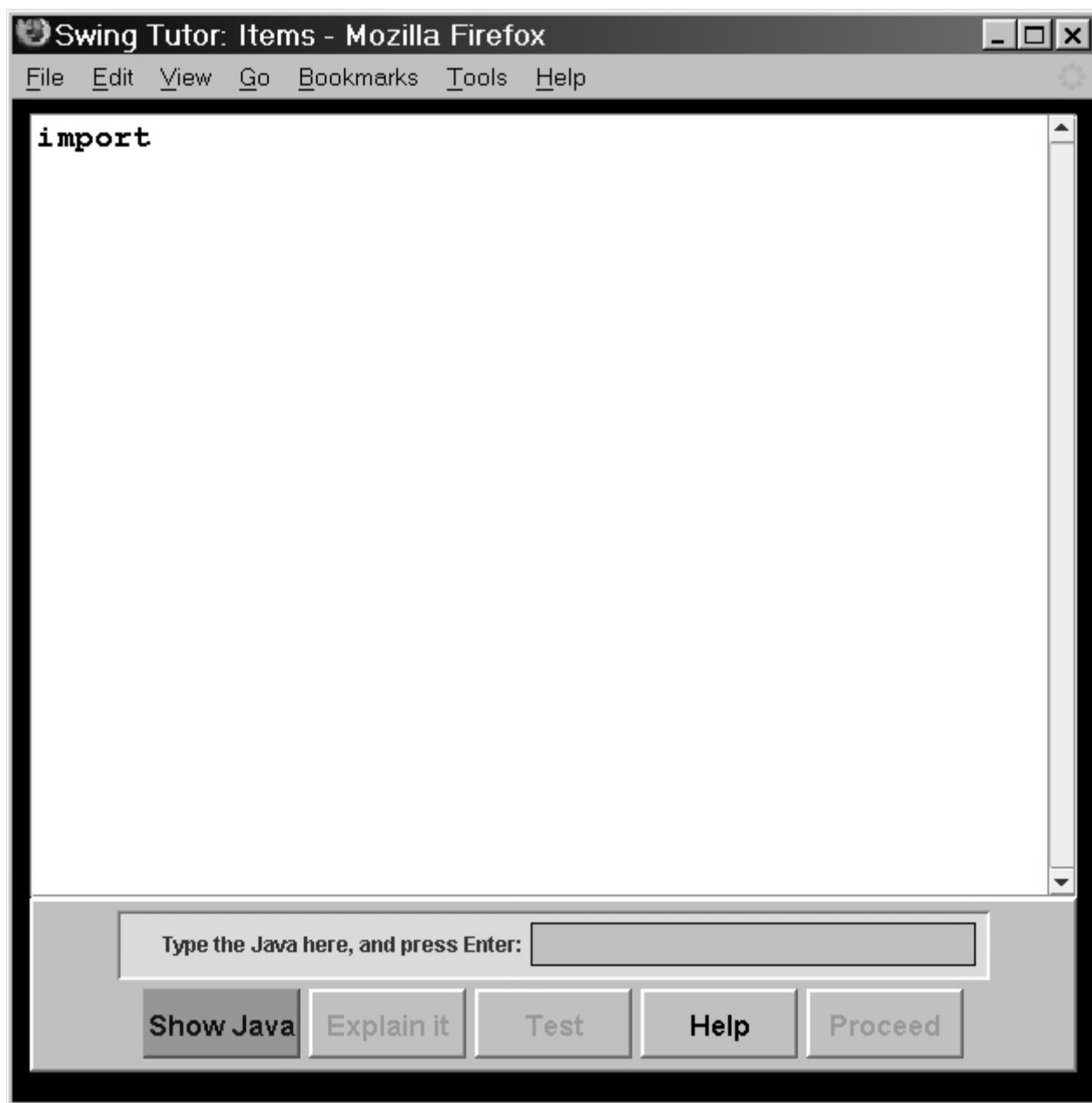


Figure 4.

Note that the name of a class, such as **JApplet**, begins with a capital letter. Also note that you **import JApplet.class** when you refer to **javax.swing.JApplet** without the dot class after **JApplet**. **These are important rules to know.** Notice also that Java is case sensitive. You will need to attend carefully to your use of uppercase and lowercase letters and other symbols.

After you use **javax.swing.JApplet** at beginning of your program, you can use **JApplet** by itself anywhere else in the program, and the compiler will know where to find the **JApplet.class** file on the system.

Here is a figure that shows the relationships between the directories and the files in them.

javax

← Directory

```
graph TD; javax[javax] --> sub1[ ]; javax --> sub2[ ]
```

Type the Java here, and press Enter:

Show Java Explain it Test Help Proceed

Figure 5.

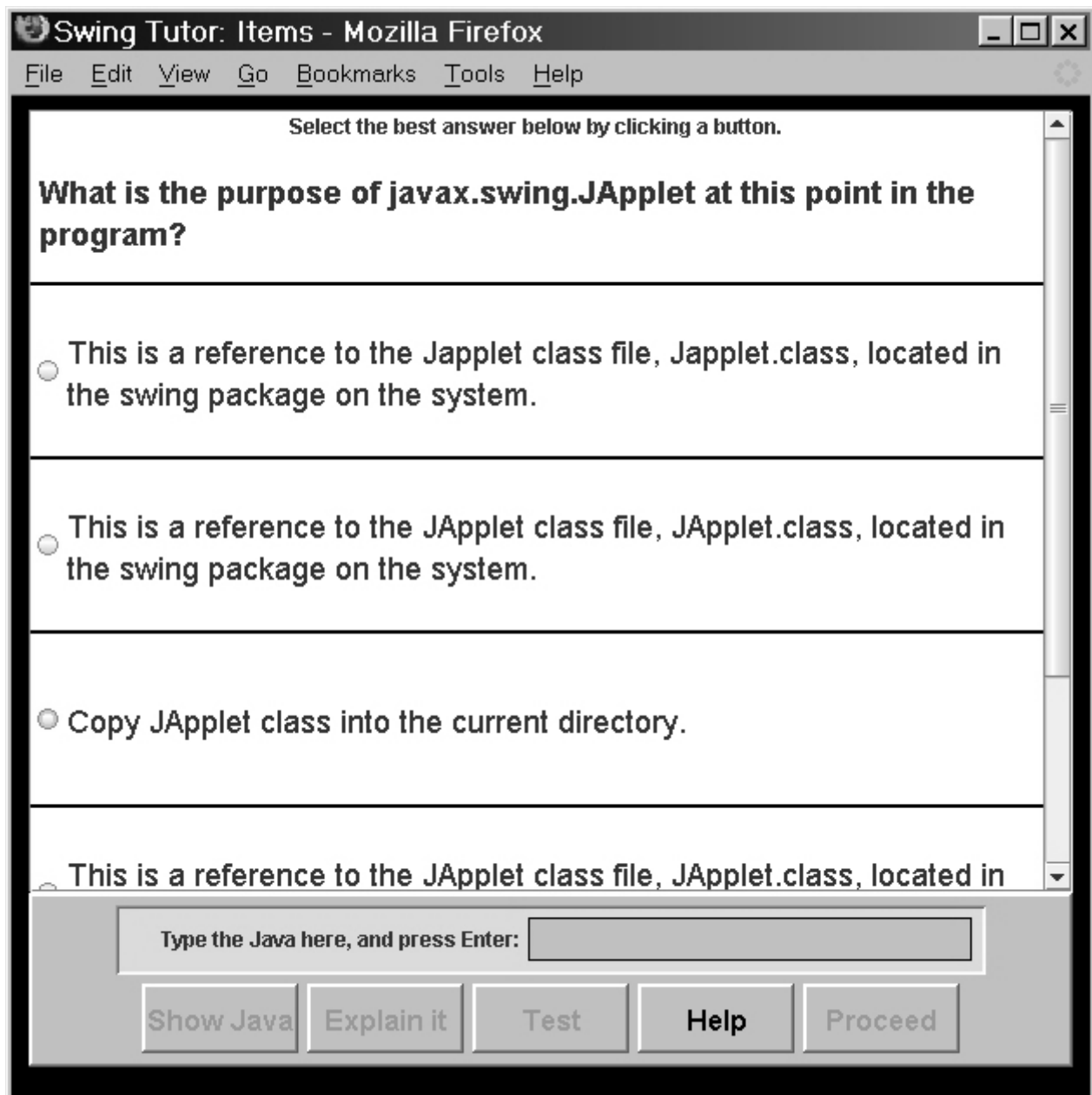


Figure 6.

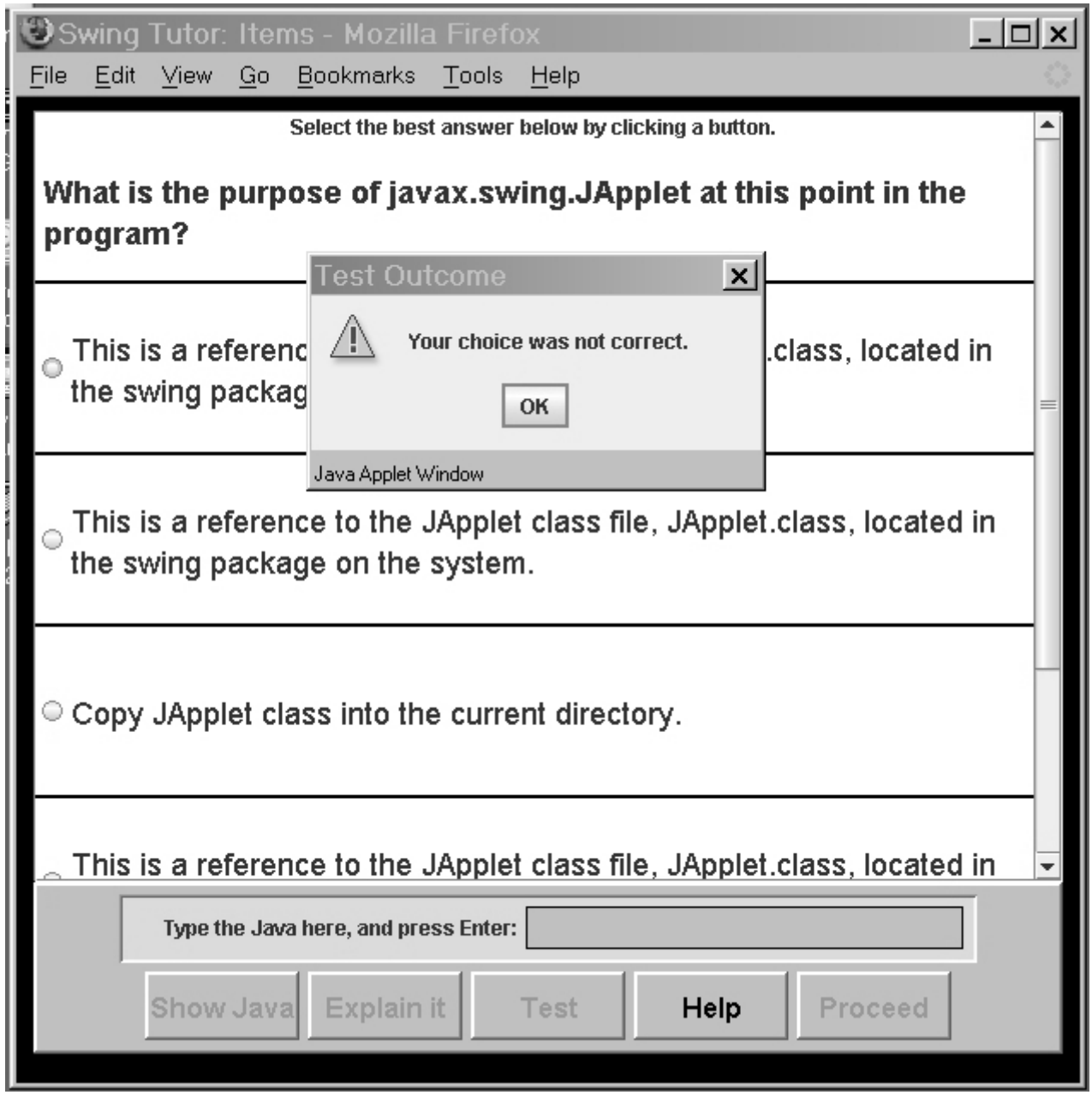


Figure 7.



Figure 8.

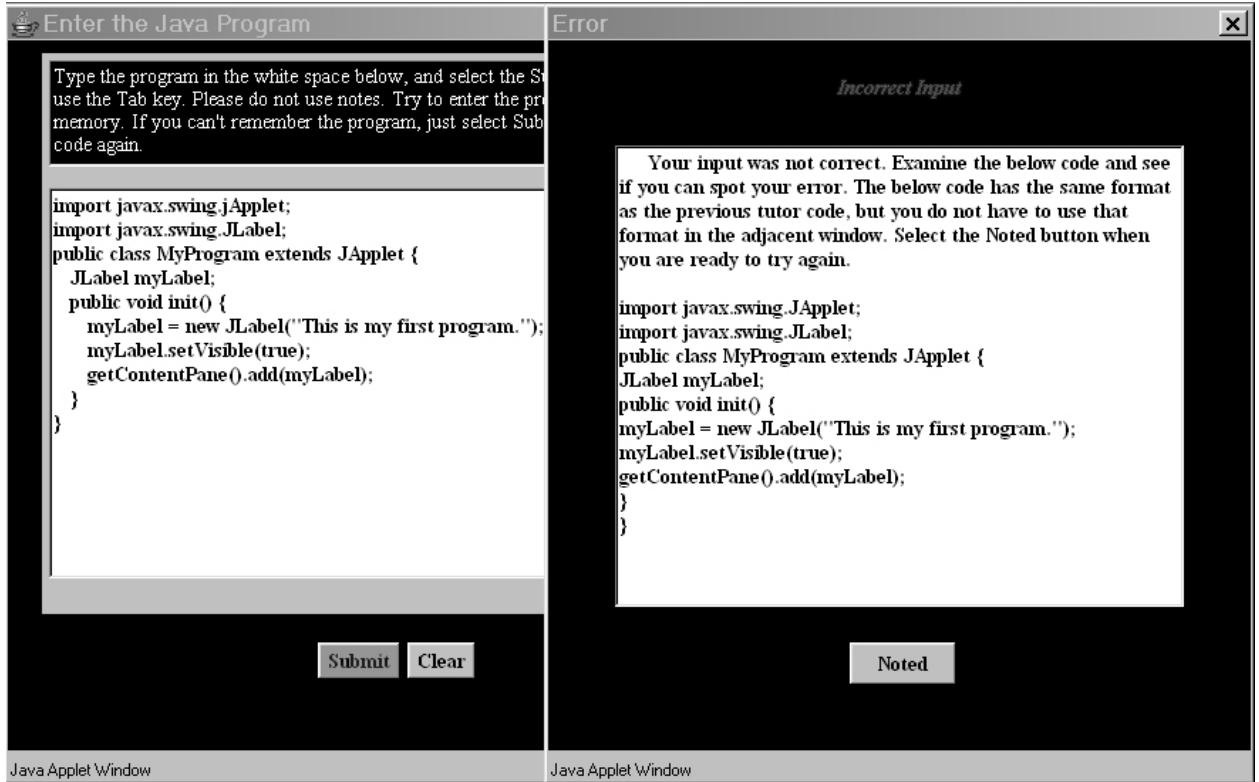


Figure 9.

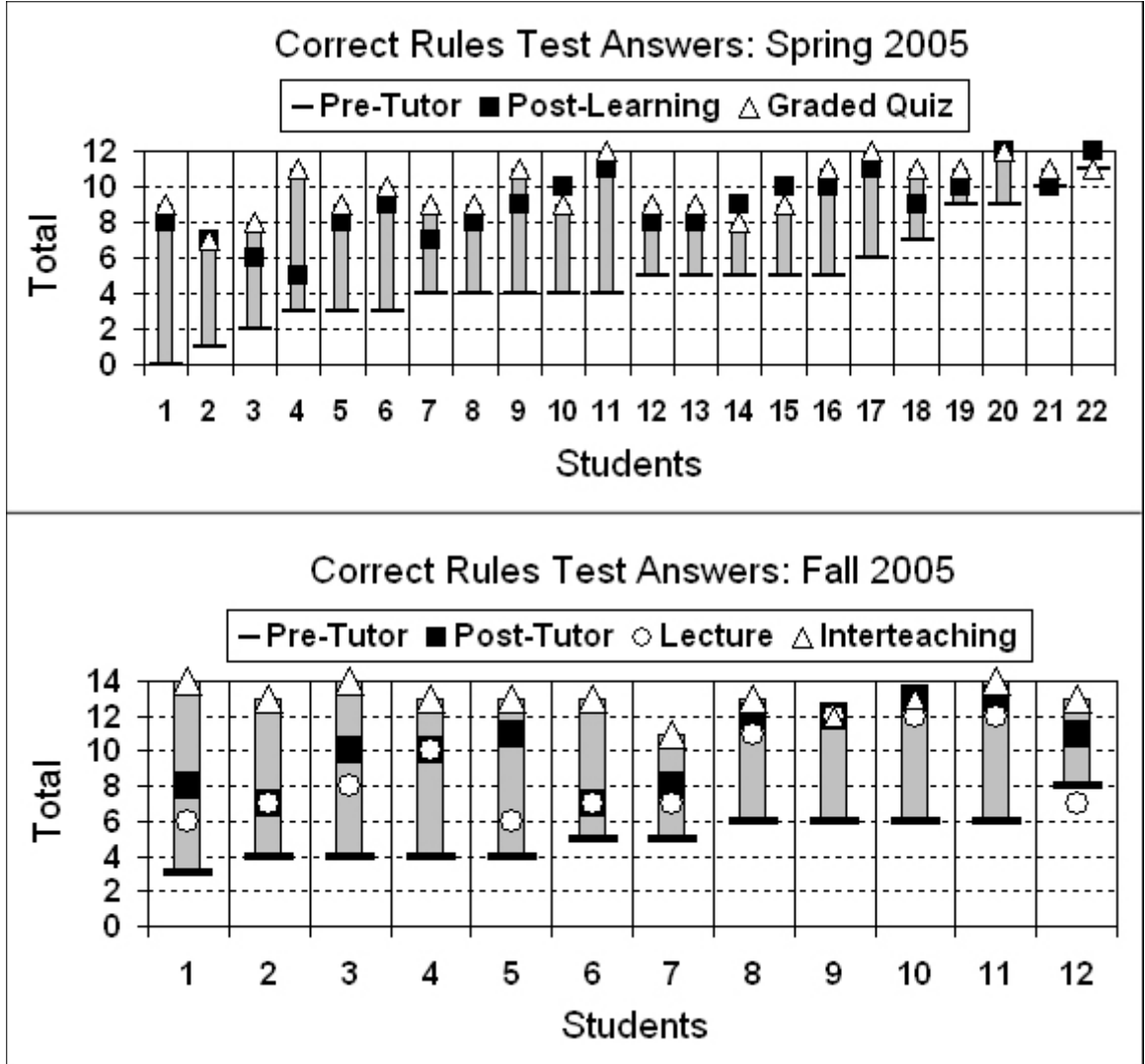


Figure 10.

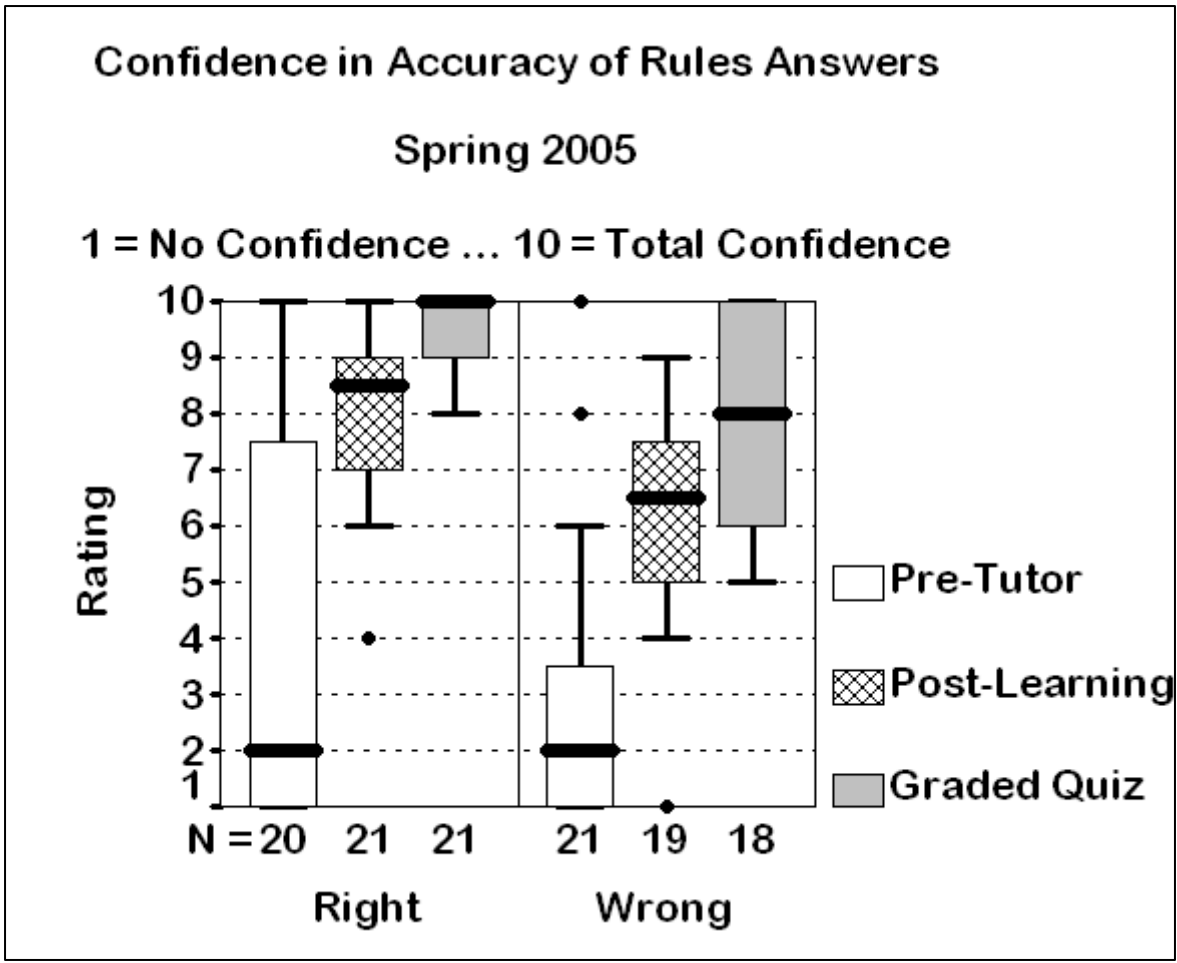


Figure 11.

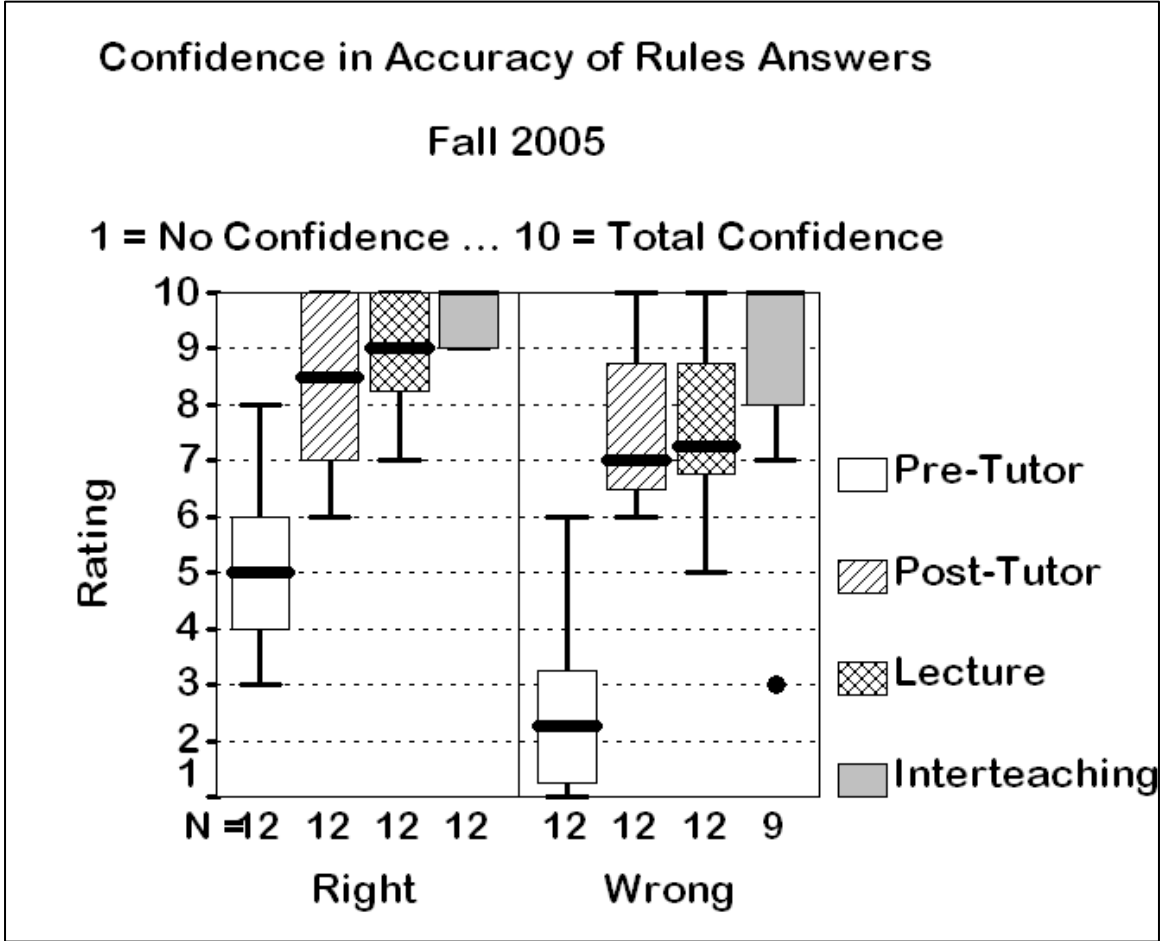


Figure 12.

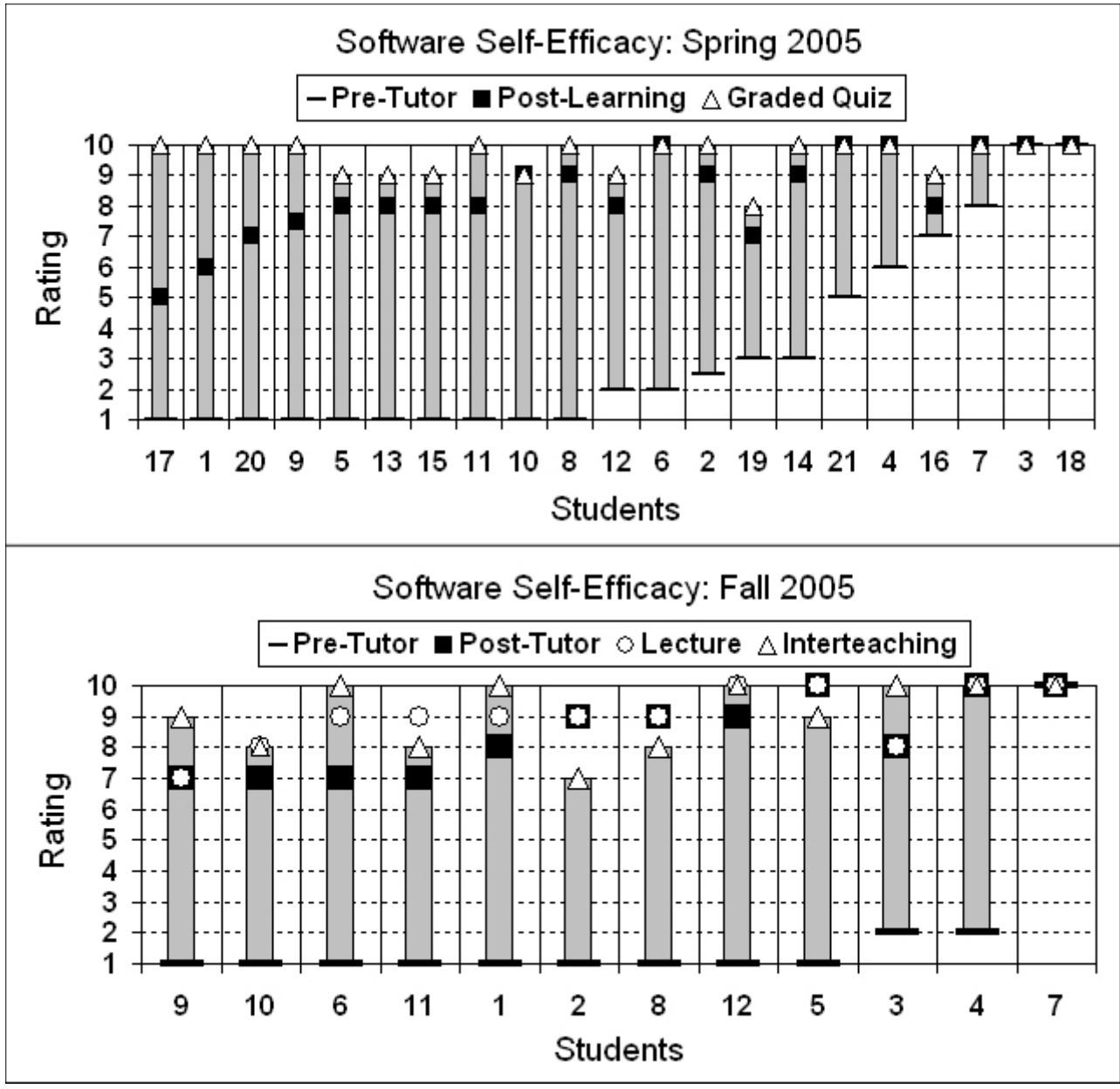


Figure 13.

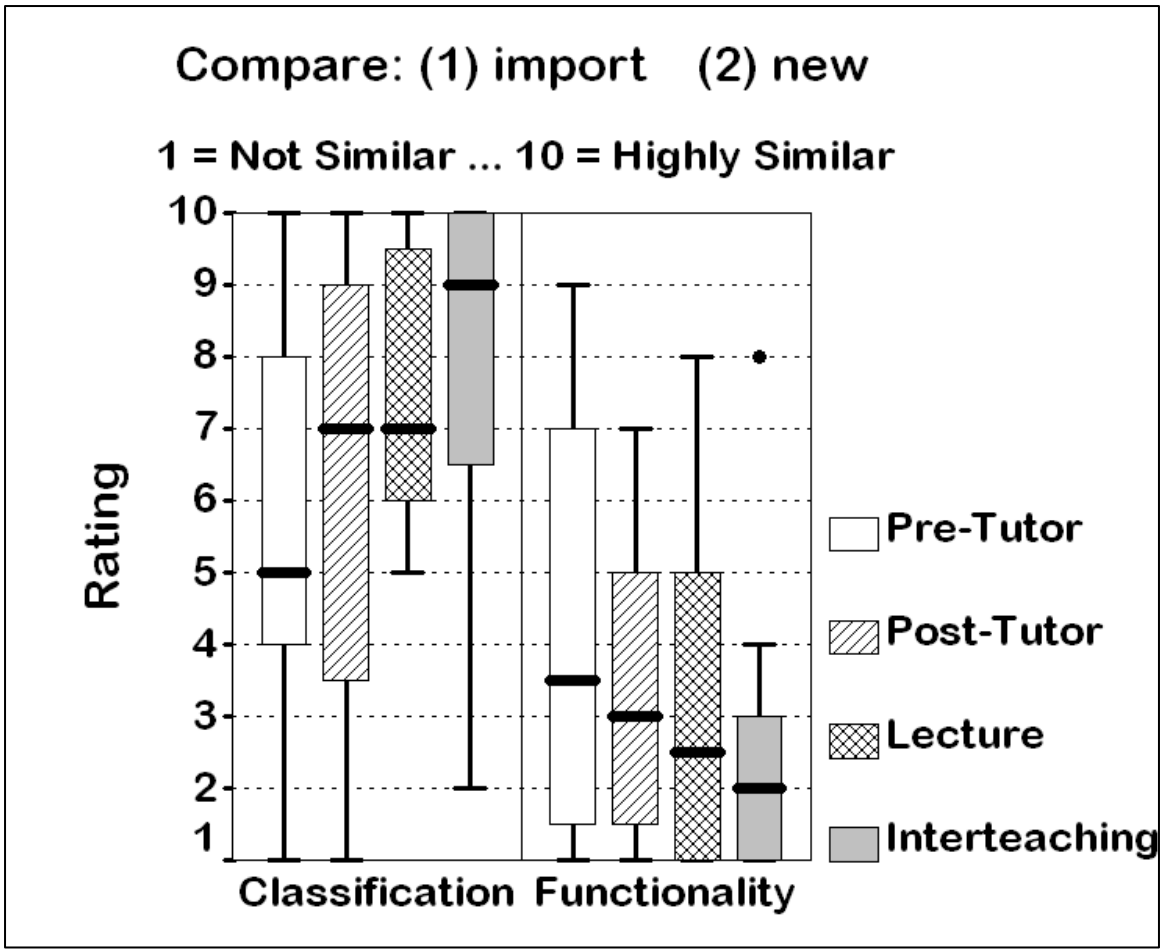


Figure 14.

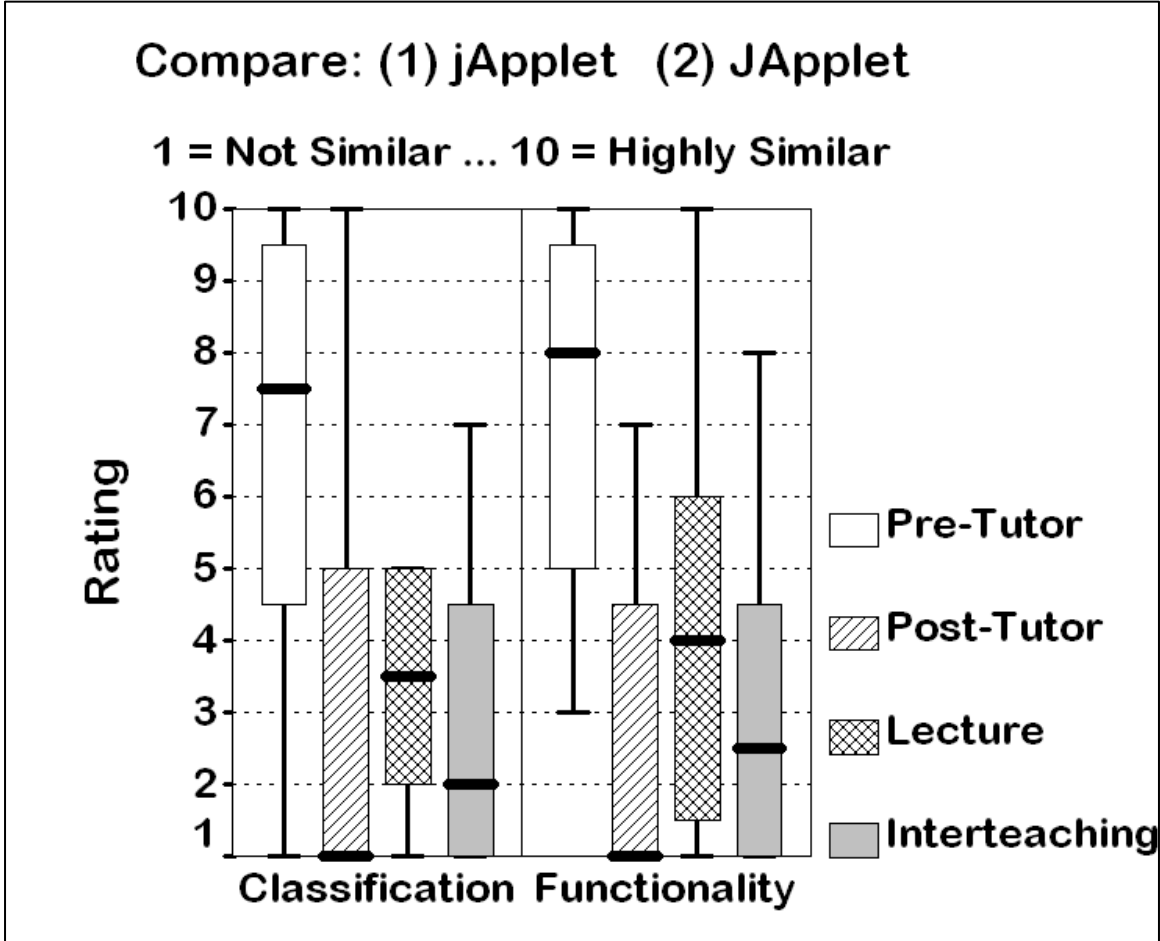


Figure 15.

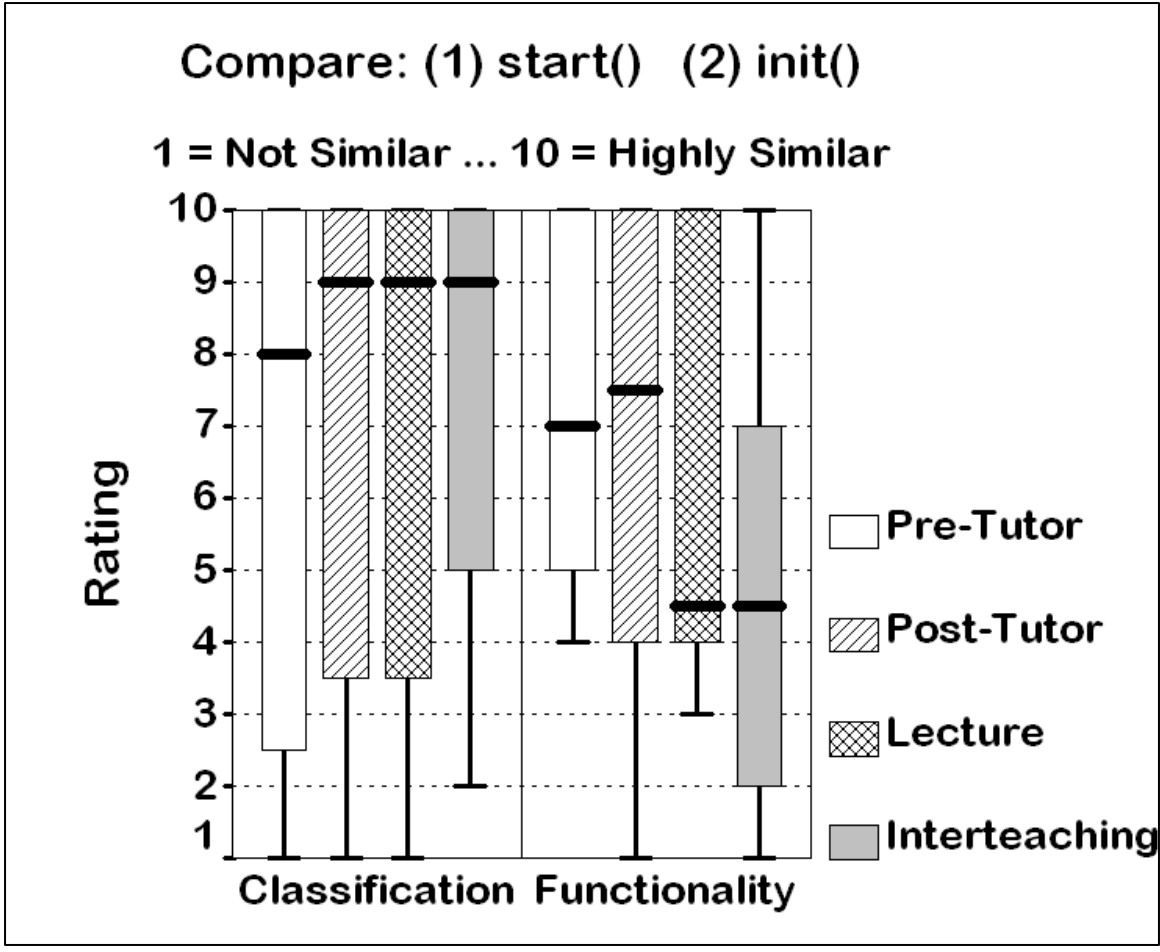


Figure 16.

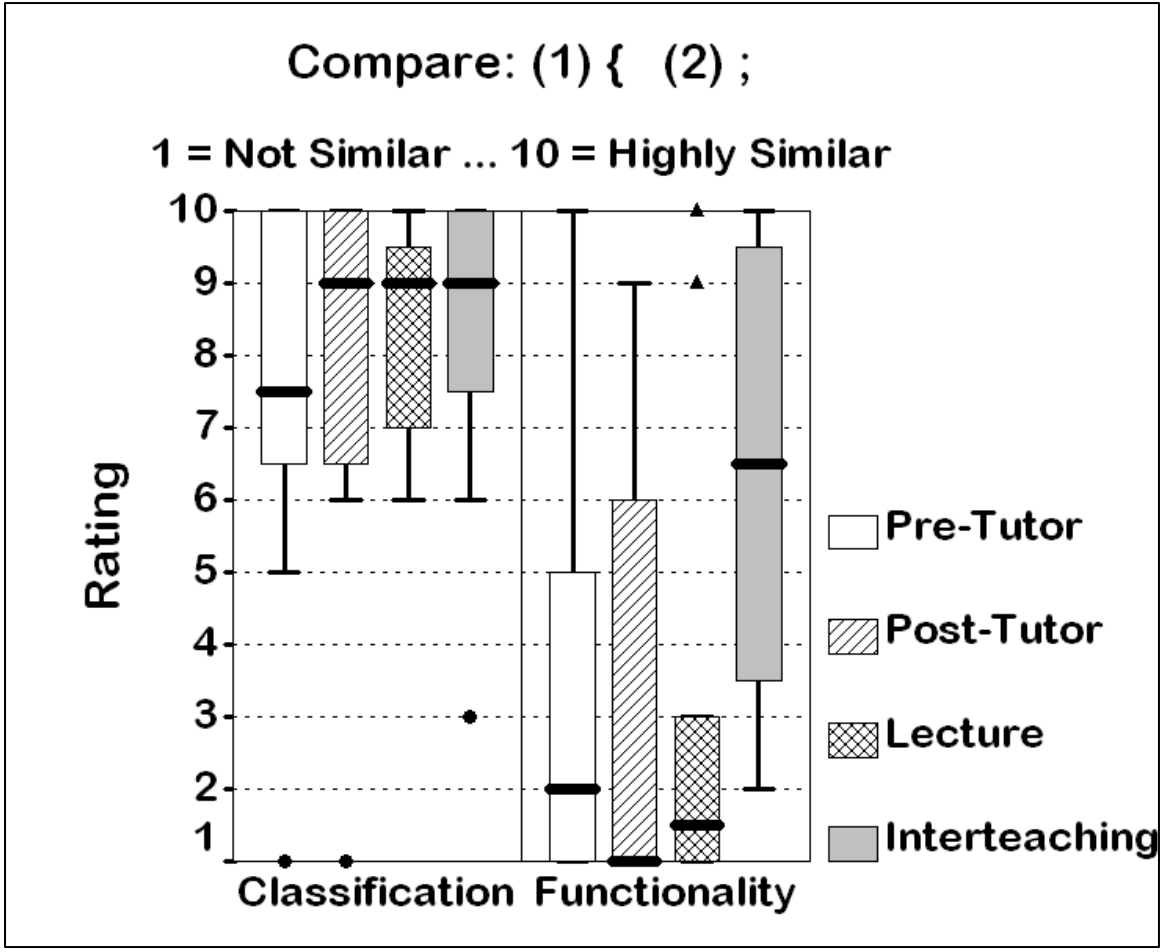
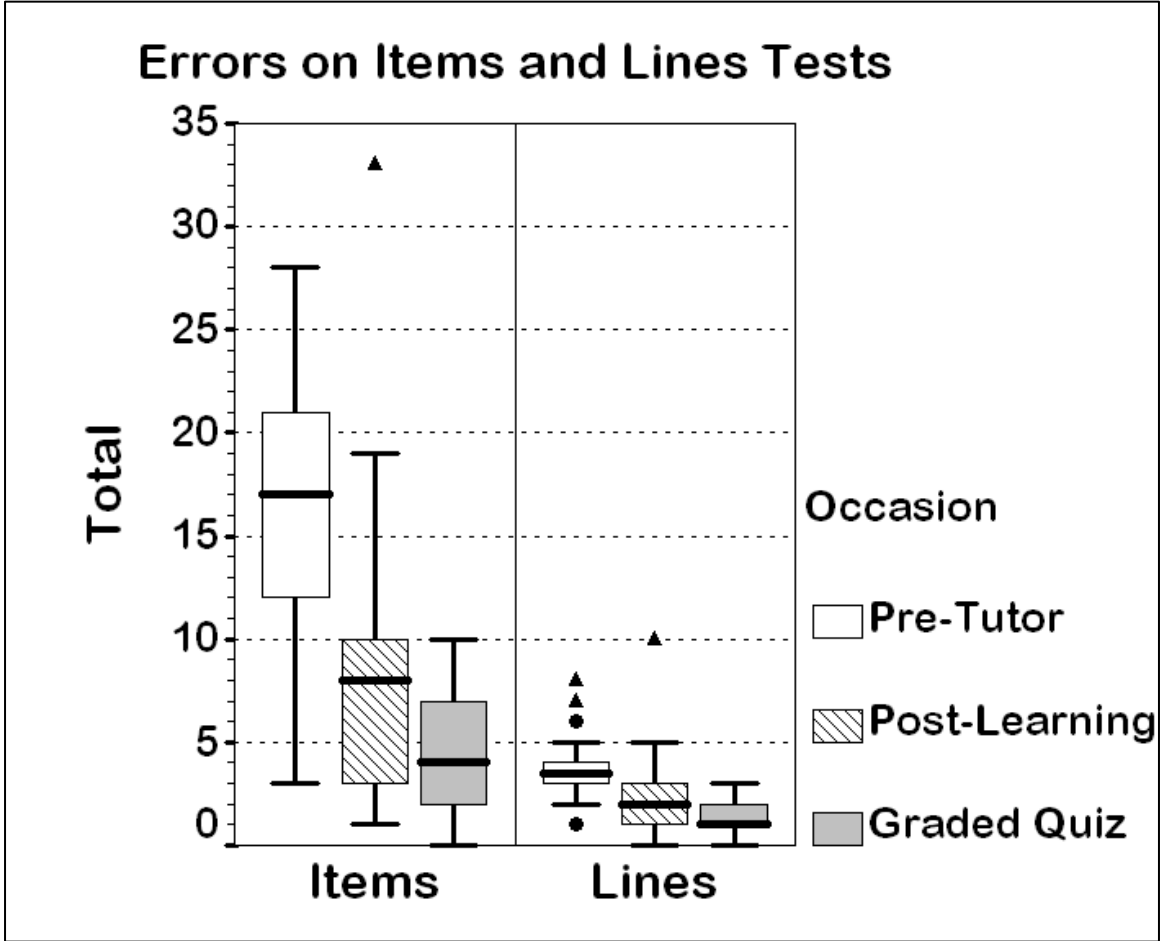


Figure 17.



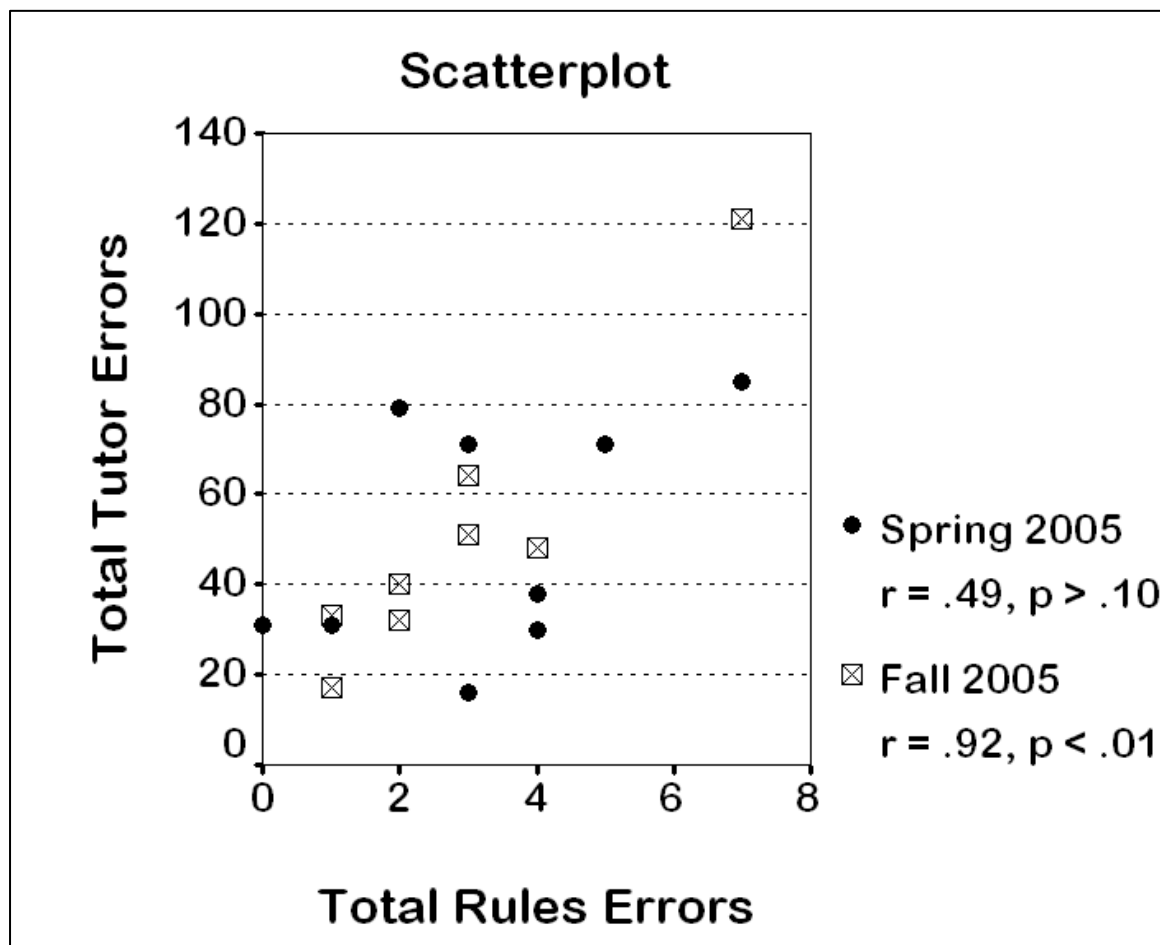


Figure 18.

Figure 19.

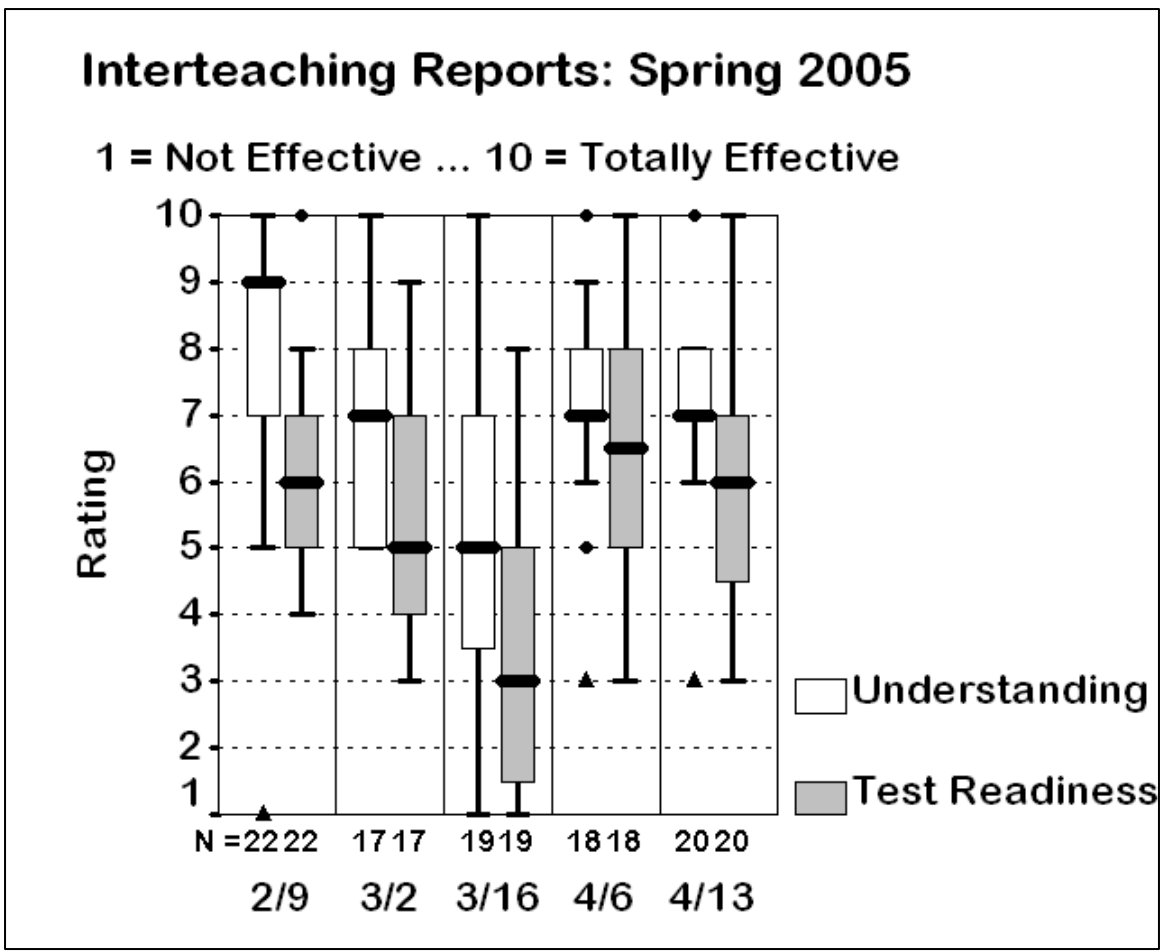


Figure 20.

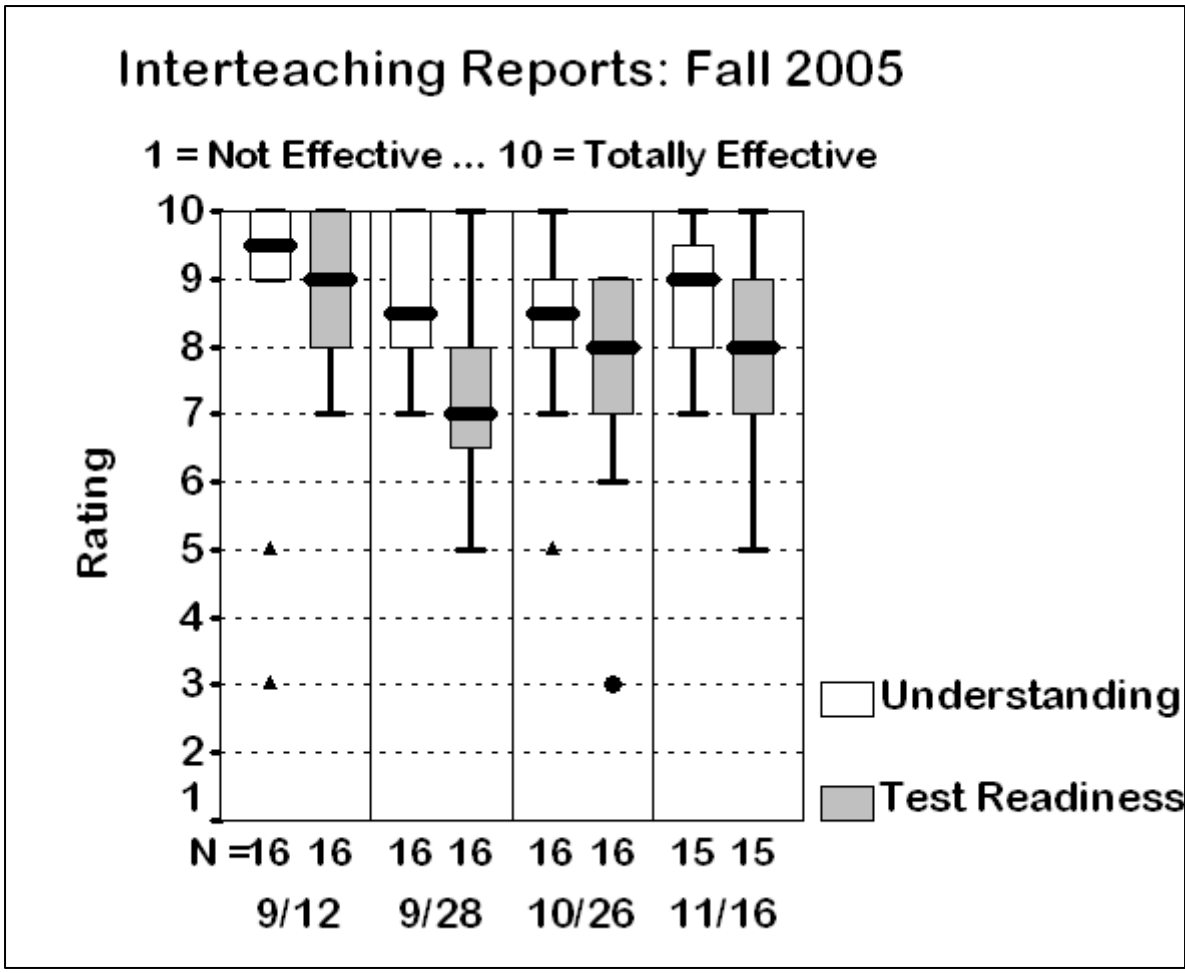
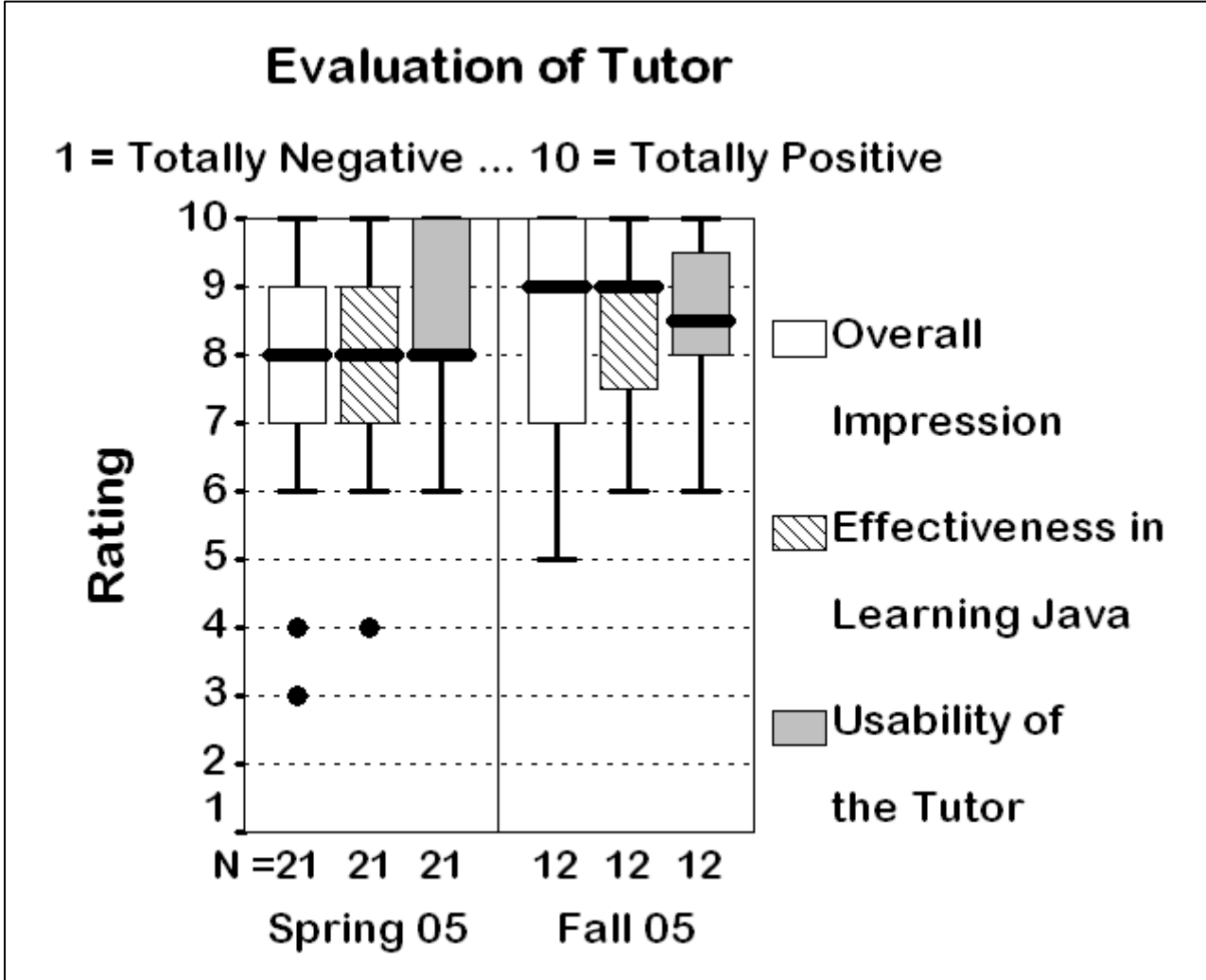


Figure 21.



BIOGRAPHIES

Henry H. Emurian is an associate professor in the Information Systems Department at UMBC. His background includes degrees in clinical psychology and computer science, and he is a licensed psychologist in Maryland. He is interested in exploring the applications of information technology to overcome problems related to education, health, and public service. His teaching and research interests include technology education strategies, Web-based tutoring systems, the empowerment of citizens via the Internet, and stress consequences of human-computer interactions. He has published in such journals as *The Journal of the Experimental Analysis of Behavior*, *Journal of E-Commerce in Organizations*, *Computers in Human Behavior*, *International Journal of Distance Education Technology*, *International Journal of Information Technology and Communication Education*, and *Aviation, Space, and Environmental Medicine*. He is a member of the American Psychological Association, the Information Resources Management Association, and the Association for Behavior Analysis. His editorial writings have appeared in the *Information Resources Management Journal*. An abbreviated version of the data for the Spring 2005 students was presented at the 2006 Hawaii International Conference on Education, Honolulu, Hawaii, January 6 – 9, 2006.

Heather K. Holden is a Ph.D. student in the Information Systems Department at UMBC. Her area of concentration is Human-Computer Interaction (HCI) with an emphasis on Technology in Education. She holds a Master of Science degree in Information Systems as well as a Bachelor of Science degree in Computer Science. An early version of the data for the Fall 2005 students was presented by Ms. Holden at the annual meeting of the Maryland Association for Behavior Analysis, Baltimore, Maryland on November 18, 2005.

Rachel A. Abarbanel is a Technical Writer for MicroStrategy, Inc. She holds a Bachelor of Arts degree in Information Systems with a minor in Economics from UMBC. In her role as a technical writer, she is responsible for managing MicroStrategy Technical Support's Knowledge Base, teaching technical support training classes, and composing various publications and documents, including a quarterly newsletter. She is interested in expanding her role in technical support to ensure that customers have all the resources they need for successful implementation of MicroStrategy applications. An early version of the data for the Spring 2005 students was presented by Ms. Abarbanel at UMBC's annual Undergraduate Research and Achievement Day on April 27, 2005.