Teaching Java™: Managing Instructional Tactics to Optimize Student Learning

Henry H. Emurian
Information Systems Department
College of Engineering and Information Technology

UMBC
1000 Hilltop Circle
Baltimore, Maryland 21250
• Among other things, I teach Java to Information Systems (IS) majors.
import javax.swing.JApplet;
import javax.swing.JLabel;
import java.awt.Color;

public class MyProgram extends JApplet {
    JLabel myLabel;
    public void init() {
        myLabel=new JLabel("This is my first program.");
        getContentPane().setBackground(Color.YELLOW);
        getContentPane().add(myLabel);
    }
}

• Near transfer (understand and recite)
• Far transfer (meaningful learning → solve novel problems)
This is my first program.
Why won’t they respond?

- In comparison to Computer Science (CS) students, Information Systems (IS) students exhibit a low rate of computer programming.
Challenges

• Students in Information Systems (IS) do not like to write computer programs.
• IS students have minimal coursework in computer programming and programming languages.
• IS students need a fundamental mastery of programming principles, especially related to the object-oriented paradigm.
• IS students are often demoralized by taking courses with computer science majors taught by computer science faculty.
• How can we best help IS students achieve the objective?
- Principles to promote retention and transfer:
  - *Repeated practice* with different instructional modalities (Halpern & Hakel, 2003)
  - *Socially supported interactions* (Fox & Hackerman, 2003)
Design-Based Research Methodology

• Observe students in context
  – Repeated observations in one classroom with one group of students and one instructor

• Improve the instructional design over successive replications
  – Systematic replication (Sidman, 1960)

• Emphasizes movement of all students to a common learning outcome (*True Gain*)
  – Contrasts with between-group studies concerned with effect size differences
Procedure

• Fall 2007 (2.5 hr Class)
  – Class 1
    • Pre-Tutor Questionnaires
    • Programmed Instruction Tutor
    • Post-Tutor Questionnaires
  – Homework
    • Prepare for Interteaching
  – Class 2
    • Lecture
    • Interteaching
      – Questionnaires
  – Class 3
    • Quiz
      – Includes Rule Test Questions from the Questionnaires

• Spring 2008 (2.5 hr Class)
  – Class 1
    • Pre-Tutor Questionnaires
    • Programmed Instruction Tutor
    • Post-Tutor Questionnaires
  – Homework
    • Brief Rule Tutor
    • Prepare for Interteaching
  – Class 2
    • Lecture
    • Interteaching
      – Questionnaires (Brief Tutor)
  – Class 3
    • Quiz
      – Includes Rule Test Questions from the Class 1 Questionnaires
Software Self-Efficacy Ratings: 23 Items

Question 4
How confident are you that you can use the following symbol now to write a Java program?

JApplet
Not at all confident. 1 2 3 4 5 6 7 8 9 10 Totally confident.
Enter a number here:

Question 5
How confident are you that you can use the following symbol now to write a Java program?

JLabel
Not at all confident. 1 2 3 4 5 6 7 8 9 10 Totally confident.
Enter a number here:

Question 6
How confident are you that you can use the following symbol now to write a Java program?

MyProgram
Not at all confident. 1 2 3 4 5 6 7 8 9 10 Totally confident.
Enter a number here:
3. Which of the following lines most likely would be used to add a JCheckBox object to a content pane?

a. getContentPane().Add(myJCheckBox);
b. container.Add(JCheckBox.Object);
c. add(container.JCheckBox);
d. getContentPane().add(myBox);
e. Add(myJCheckBox);

Enter a letter here:

How confident are you that you selected the correct answer? Not at all confident.  1     2     3     4     5     6     7   8     9     10   Totally confident.

Enter a number here:
7. Which one of the below lines declares mySlider as a potential instance of the JSlider class?

a. JSlider mySlider;
b. JSlider myJSLider;
c. mySlider JSlider;
d. myJSlider JSlider;
e. JSlider mySlider.

8. Given the following line in a program, `public class MyJLabel extends JLabel`, which one of the below would be the name of the file that contains this program for compilation?

a. MyJlabel.java
b. MyJLabel.java
c. MyJLabel.javax
d. myJLabel.java
e. JLabel.java
What Instructional Modalities Make Sense?
**Programmed Instruction**

1. A set of **structured interactions** between a learner and a tutor.
2. Occasions **disciplined study behavior** that is focused on the individual learner.
3. Manages the **moment-by-moment interactions** between a learner and a tutor: *learn units*.
4. A **step-wise progression** from elementary facts to the achievement of meaningful learning.

---

**Java Tutor**

[http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/SwingTutorLinksV2.html](http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/SwingTutorLinksV2.html)
Java Tutor: Item Learning

As a reminder, here is the program that you will learn to write and understand:

```java
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 do not need instructions to use this tutor because the events are determined by the enabled buttons and by the accuracy of your typed input and other selections that you will make. If you can't read the Java item when asked to type it, simply press the Enter key with the cursor in the JApplet area.

The second line of code is this:

```java
import javax.swing.JLabel;
```

The Java term being taught is highlighted in blue.

```java
import javax.swing.JLabel;
```

The `import` term, when used in the above line, allows you to refer to the built-in Java class `JLabel`, with a shorthand notation. The reason to use the `import` keyword is because the `JLabel` class file is not located in your current directory, and it is needed to write your program.

For example, to use the built-in Java class file `JLabel`, in your program, you write:

```java
import javax.swing.JLabel;
```

At the beginning of the program, then you may use `JLabel` by itself in the program, as explained later in the tutor, rather than having to use `javax.swing.JLabel` in the program. The compiler will then be able to find the `JLabel` class file on the system. The `import` keyword then:

Which one of the following statements is correct?

- `import java.swing.JApplet;`
- `import java.swing.JApplet;`
- `import java.swing.JLabel;`
import javax.swing.JApplet;
import javax.swing.JLabel;
import java.awt.Color;

Your choice was correct.

Java has many keywords, and you will learn about them as you progress in your study and use of the language. Keywords are reserved words, and they can not be used by the programmer for other applications, such as a variable name. The public term, then, may be classified as a keyword in the Java programming language.
The expression `javax.swing.JLabel` refers to a file named `JLabel.class`. This is a file that is used to construct instances of the `JLabel` class. Since your program will construct an instance of the `JLabel` class, it is easier to use a shorthand notation to refer to the `JLabel.class` file.

When you write

```java
import javax.swing.JLabel;
```

at the beginning of the program, this allows the later use of the built-in `JLabel` file by the shorthand notation, which is `JLabel`, by itself.

The `JLabel` class file, `JLabel.class`, as it exists in a directory, contains the compiled Java code to display text on the screen, and it is located in the `javax.swing` package on the system. You can think about a package as a directory in which related files are stored. The reason that you have to use either `import` or the full path, `javax.swing.JLabel`, is simply because the `JLabel.class` file is not in your directory. That file is in a different directory on the system, and the compiler needs to know where it is located before it can be used in your program.

Notice that `JLabel` begins with a capital letter. That tells you that it is a class file in Java. That is an important rule to know. The definition of a class is presented later.
Type the program in the white space below, and select the Submit button. Do not use the Tab key. Please do not use notes. Try to enter the program from your memory. If you can't remember the program, just select Submit, and you can see the code again.

```java
import java.awt.JApplet;
```

Your input was not correct. Examine the below code and see if you can spot your error. The below code has the same format as the previous tutor code, but you do not have to use that format in the adjacent window. Select the Noted button when you are ready to try again.

```java
import java.awt.JApplet;
import java.awt.Label;
import java.awt.Color;
public class MyProgram extends JApplet {
    JLabel myLabel;
    public void init() {
        myLabel = new JLabel("This is my first program.");
        getContentPane().setBackground(Color.YELLOW);
        getContentPane().add(myLabel);
    }
}
```
1. **Lecture**
   - Repeat the tutor material while students write the code

2. Run the applet on the web
1. A mutually probing, mutually informing conversation between two people (Boyce & Hineline, 2002)

2. The questions on a topic to be addressed by the participants during a dialogue are prepared in advance by the teacher, and the students come prepared to interteach

3. Has the objective of insuring, by the participants as a team, that each member of the dyad can answer the questions with understanding
Preparation for Interteaching: Brief Rule Tutorial

http://userpages.umbc.edu/~emurian/learnJava/swing/tutor/v2/rules/Tutor.html
Interteaching is a Collaboration Session with Two Students Participating

Interteaching Objectives

Before the next class meets, you must complete the brief Rules Tutorial for credit (20 points). The tutorial may take 30 minutes to complete. You may repeat the tutorial as often as you want. The link to the Rules Tutorial is given in the Assignments folder on the course Bb site.

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

You should prepare for the interteaching session to 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.

```java
import javax.swing.JApplet;
import javax.swing.JLabel;
import java.awt.Color;
public class MyProgram extends JApplet {
    JLabel myLabel;
    public void init() {
        myLabel = new JLabel("This is my first program.");
        getContentPane().setBackground(Color.YELLOW);
        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 an operator? Give an example.
5. What is a keyword? Give an example.
6. What is an identifier?
7. What does it mean that methods may be inherited from a superclass?
8. What is the meaning of override?
9. How can you identify a series of characters as the name of a method?
10. What is a constructor method? What properties of the syntax make it a constructor method?
11. Describe the position and functions of the terms in a statement that uses a method to change a property of an object.

During the interteaching session, you may have access to the explanations of items that were presented in the tutor:

http://userpages.umbc.edu/~emurian/learnJava/swing/tutor/v2/explanations/Explanations.html

You also may have access to the explanations of the rules in the brief tutorial:

http://userpages.umbc.edu/~emurian/learnJava/swing/tutor/v2/rules/explanations/

Interteachers in Action
A Mann Whitney U test was marginally significant for reported Java programming experience between the two classes ($Z = -1.933, p = 0.053$).
Tutor Evaluation

1 = Totally Negative ... 10 = Totally Positive

Scale
- Overall
- Learning
- Java
- Usability

Rating

Fall 2007 (n = 17)  Spring 2008 (n = 16)

Class
Software Self-Efficacy

1 = No Confidence ... 10 = Total Confidence

Occasion
- Pre-Tutor
- Post-Tutor
- Interteaching

Rating

Class

Fall 2007 (n = 17)    Spring 2008 (n = 16)
Interteaching Evaluation

Interteaching Evaluation: Fall 2007
1 = Not Effective ... 10 = Totally Effective

Interteaching Evaluation: Spring 2008
1 = Not Effective ... 10 = Totally Effective
Multiple-Choice Test Errors: Tutor → Quiz

**Fall 2007**

- **Items:** $r = 0.551, p = 0.022$
- **Rows:** $r = 0.039, p = 0.881$

**Spring 2008**

- **Items:** $r = 0.837, p = 0.000$
- **Rows:** $r = 0.649, p = 0.007$

$n = 17$ for Fall 2007

$n = 16$ for Spring 2008
Correct Rule Test Answers: Individuals

Correct Rule Test Answers: Fall 2007

Correct Rule Test Answers: Spring 2008
Rule Test Errors

Rule Test Errors: Fall 2007 (n = 17)

Rule Test Errors: Spring 2008 (n = 16)
History Counts…

Fall 2007 (n = 17)

Spring 2008 (n = 16)

R Sq Linear = 0.068

R Sq Linear = 0.51

R Sq Linear = 0.501

R Sq Linear = 0.588
History Counts...

Fall 2007 (n = 17)

R Sq Linear = 0.018

Self-Reported Java Experience

Tutor Item Errors

Spring 2008 (n = 16)

R Sq Linear = 0.285

Self-Reported Java Experience

Tutor Item Errors

Fall 2007 (n = 17)

R Sq Linear = 0.002

Pre-Tutor Software Self-Efficacy

Tutor Item Errors

Spring 2008 (n = 16)

R Sq Linear = 0.405

Pre-Tutor Software Self-Efficacy

Tutor Item Errors
Software Self-Efficacy & Tutor Items Errors

- Fall 2007
- Spring 2008
- Linear (Fall 2007)
- Linear (Spring 2008)

Fall 2007: $R^2 = 0.0025$
Spring 2008: $R^2 = 0.4047$
History Counts…

Fall 2007 (n = 17)

Spring 2008 (n = 16)

R Sq Linear = 0.107

R Sq Linear = 0.37

R Sq Linear = 9.81E-4

R Sq Linear = 0.358

Self-Reported Java Experience

Self-Reported Java Experience

Pre-Tutor Software Self-Efficacy

Pre-Tutor Software Self-Efficacy
History Counts…

Rule Test Errors: Fall 2007 (n = 17)

R Sq Linear = 0.013

p = 0.685

Rule Test Errors: Spring 2008 (n = 16)

R Sq Linear = 0.327

p = 0.021
How History Counts…

Fall 2007 (n = 17)

Spring 2008 (n = 16)

\[ R^2 \text{ Linear} = 0.303 \]

\[ R^2 \text{ Linear} = 0.701 \]

\[ p = 0.022 \]

\[ p = 0.000 \]
History Counts…

Rule Test Errors: Fall 2007 (n = 17)

R Sq Linear = 0.223

p = 0.055

Rule Test Errors: Spring 2008 (n = 16)

R Sq Linear = 0.494

p = 0.002
Self-Reports of Confidence in Answers

Confidence in Rule Test Answers: Fall 2007

Confidence in Rule Test Answers: Spring 2008
Correct Rule Test Answers: Fall 2007

Correct Rule Test Answers: Spring 2008

Pre-Tutor, Post-Tutor, Interteaching, Quiz

$r = 0.98$, $n = 8$, $p = .000$

$r = 0.87$, $n = 9$, $p = .002$
Classify: (1) import  (2) new

1 = Not Similar ... 10 = Highly Similar

Occasion
- Pre-Tutor
- Post-Tutor
- Interteaching

Class
- Fall 2007 (n = 17)
- Spring 2008 (n = 16)
Classify: (1) myLabel  (2) JLabel

1 = Not Similar ... 10 = Highly Similar

Occasion
- Pre-Tutor
- Post-Tutor
- Interteaching

Class
Fall 2007 (n = 17)  Spring 2008 (n = 16)
Classify: (1) getContentPane()  (2) init()

1 = Not Similar ... 10 = Highly Similar

Occasion

- Pre-Tutor
- Post-Tutor
- Interteaching

Rating

Class

Fall 2007 (n = 17)  Fall 2008 (n = 16)
Classify: (1) . (2) ;

1 = Not Similar ... 10 = Highly Similar

Occasion
- Pre-Tutor
- Post-Tutor
- Interteaching

Rating

Class
- Fall 2007 (n = 17)
- Spring 2008 (n = 16)
Classify: (1) `javax.swing.JLabel` (2) `java.awt.Color`

1 = Not Similar ... 10 = Highly Similar

Class

Fall 2007 (n = 17) Spring 2008 (n = 16)

Rating

Occasion
- Pre-Tutor
- Post-Tutor
- Interteaching
Classify: (1) YELLOW  (2) Color

1 = Not Similar … 10 = Highly Similar

Occasion
- Pre-Tutor
- Post-Tutor
- Interteaching

Fall 2007 (n = 17)  Spring 2008 (n = 16)

Class

Rating
Classify: \( (1) = (2) \) \\

\[1 = \text{Not Similar} \ldots 10 = \text{Highly Similar}\]
Classify: (1) MyProgram  (2) JApplet

1 = Not Similar ... 10 = Highly Similar

Occasion
- Pre-Tutor
- Post-Tutor
- Interteaching

Rating

Fall 2007 (n = 17)  Spring 2008 (n = 16)

Class
Challenges with Programmed Instruction

• It is labor intensive to develop.
  – We have proposed to develop a generic shell.

• There are conceptual issues regarding the size of a learn unit.
  – The opportunity for repetition, until a multiple-choice item is answered correctly, can lead to careless reading.
Challenges with Interteaching

• A rare student will show an aversion to collaborative learning.
• Pairs of students need different amounts of time.
• It is difficult to assess the “quality” of a collaboration objectively.
• “Understanding” is more than an intraverbal performance.
Challenges with Lecture

- I have to know what I’m talking about.
Conclusions

1. Programmed instruction is an effective tool in technology education.
   • It meets the needs of the individual learner.
   • The instructional design can promote meaningful learning and self-confidence.
   • The tutoring system is well-received by novitiate learners.
2. Interteaching may add value, but there are issues of retention and transfer.
3. The competency attained sets the occasion for advanced learning with enthusiasm.
4. Students like the tutor and the interteaching, and so do I.
5. I also like to use lectures with hands-on learning and classroom collaboration among students.
Thank you!

Questions?

The tutor, the source code, and all instructional material are freely available on the web.