

A Structured, Comparative, Module-Based Approach for Teaching Java in Data Structures Courses

Kenneth E. Martin
Department of Computer and Information Sciences
University of North Florida
4567 St. Johns Bluff Road S.
Jacksonville, Florida 32224

Abstract - This paper describes an approach to teaching data structures to computer science and/or information systems majors where Java is the language used, but it was NOT the initial language used in the introductory programming course (neither for computer science students nor for information systems students). In the former case, the students entered this course being familiar with the C language (two courses) while in the latter case they were familiar with the COBOL language (two courses) and sometimes C.

Students studied twenty-seven specially constructed and focused learning modules, which were posted to the WWW, so that they could quickly learn the essentials of Java, enabling the instructor to spend the vast majority of the time in the course on the data structures themselves. Each module was designed to focus on fundamental concepts with a strong emphasis on enabling students to quickly find and analyze the appropriate Java packages that were necessary. No emphasis was placed on applets (except for visualization of data structures), so students would learn to program applications taught in typical, traditional data structures courses.

Students provided electronic (using the WWW) responses to the same ten statements regarding each of the twenty-seven modules plus a summary of all the modules at the end of the course; open-ended comments were permitted and encouraged. In general, the effectiveness of placing the Java API documentation, the code itself, and the run from the code in close proximity was hypothesized to be an effective teaching strategy, and indeed this turned out to be the case.

Students evaluated each module for effectiveness in helping to learn Java, for appropriate internal documentation, for effectiveness of the Java API documentation, for the level of complexity of the module, for the usefulness of the module in similar situations, and for the current student attitude toward learning Java. In addition, for each module, the ease of learning Java was compared to the ease of learning either C or COBOL. Students were specifically

encouraged NOT to buy a Java textbook, but rather to simply study and analyze the modules themselves. Students appeared to be extremely comfortable and pleased with this approach.

The instructor analyzed student perception of each of the modules seeking an item-by-item statistical comparison of computer science and information systems student reactions, and also compared student perceptions over the timeline allocated to the modules and at the end of the full course. In general, students with a background in COBOL appeared to be more favorably disposed to learning Java for data structures than students with a background in C, although both strongly favored this approach to using Java for data structures. In fact, based upon this experiment, the information systems curriculum was recently modified to require an object-oriented language (currently it would be Java) for the data structures course.

Introduction

The purposes of this paper are to:

- describe a common web-based set of twenty-seven Java Modules (JM) for introducing Java in data structures courses to computer science and information systems students (different courses) for whom, in most cases, there was no previous knowledge of Java or any other object-oriented programming language;
- describe the differences in emphases between the two courses;
- determine the most important aspects of the modules;
- describe and measure student reactions to the JM;
- compare and contrast computer science vs. information systems student reactions to the JM; and
- describe and measure student reactions to the ease of learning Java versus C or Cobol.

The Modules

The author produced twenty-seven modules covering the fundamental concepts of programming in Java. Each was built around a single program, and because the typical data structures course does NOT use visual

programming, the modules did NOT cover applets. The basic premise was that the essentials of Java could be taught in the first four weeks of a fifteen-week semester without using a textbook, and regardless of whether the students had a background in C (computer science students) or Cobol (information systems students). Perhaps object-oriented program can actually be taught in CS1 [1] and in fact we are now going to replace C with Java for Information Systems students.

Topics for the various modules are given below (documentation necessary for a particular package might appear in several modules if it was necessary for an application). This iterative and spiral approach might be described as “gentle” [3] and certainly the author tried to emphasize patterns viz a viz the documentation and the overall thought process of design [5].

1. Show documentation for `java.lang.*`, `java.io.*`, `java.lang.System`, and `java.io.PrintStream`. These were used to construct a car, and then show run and disassembled bytecodes using the `-c` option for `javap`.
2. Show documentation for `java.util.*`, `java.util.Date`. These were used to construct a Date and show run.
3. Do arithmetic tests and show run (C-like concepts).
4. Show documentation for `java.lang.Short`, `java.lang.Integer`, and `java.lang.Float`. Do pre and post-arithmetic tests and show run.
5. Show documentation for `java.awt.*` and `java.awt.Point`. Construct a point and show run.
6. Show documentation for `java.lang.String`. This was used to study strings and show run.
7. Show documentation for `java.awt.Point` and then use reference variables and show run.
8. Show documentation for `java.lang.String` and the use object references and object values and show run.
9. Show documentation for `java.lang.reflect.*` and `java.util.Random` and use them to see the methods of a class and show run.
10. Show documentation for `java.lang.String` and use for an array of Strings (with run).
11. Check numbers for divisibility by 7 and show run.
12. Convert digits to strings using cases and show run.
13. Show documentation for `java.lang.String`, use a for loop with an array of Strings, and show run.
14. Construct arrays of ints and floats with casts, use while loop, and show run.
15. Use the do-while statement and show run.
16. Use the break statement with a for statement and show run.
17. Construct an object with default constructor, send a message, and show run.
18. Study the scope of a variable and show run.
19. Study the concept of pass by value and show run.
20. Study command line arguments and show run.

21. Show the documentation for `java.lang.Integer` including `MAX_VALUE` and `MIN_VALUE` and show run.

22. Show the documentation for `java.awt.Point` and use it to make rectangles with overloaded methods and show run.

23. Show constructor methods for a person and show run.

24. Show the documentation for `java.awt.Point`, overload constructors for a rectangle, and show run.

25. Show the documentation for `java.lang.Object` and `java.lang.Class`, get the name of a class and extend a class and use its method (show run).

26. Extend a class, override its method, and show run.

27. Show the documentation for `java.awt.Point`, extend `java.awt.Point` and make a constructor (show run).

28. Summarize your view of all the modules.

The Data Structures Emphasized In The Two Courses And Their Common Java Emphasis

The course for computer science students was a 3rd course, where they previously had taken Introduction to C and Data Structures Using C. The current course was entitled Advanced Data Structures and focused heavily on the concept of an interface (and abstract classes) and the resultant separation of the abstraction from the implementation [7]. Fundamental data structures were quickly reviewed and considerable time was spent on such topics as graphs, randomization [2], file compression, word-search puzzles, alpha-beta pruning, simulation, iterators, AVL trees, red-black trees, AA-trees, B-trees, hashing, priority queues, splay trees, pairing heap operations, and the disjoint set class.

The course for information systems students was also a 3rd course, where they previously had taken Introduction to Cobol and File Structures Using Cobol. Many also had taken Introduction to C, but it was not required. The current course was titled Data Structures and focused on arrays, simple sorting, stacks and queues, linked lists, recursion, advanced sorting, binary trees, hash tables, heaps, graphs, and weighted graphs [4].

In both cases, the emphasis in the current course was to learn enough Java applications so that the appropriate data structures (advanced or fundamental) could be programmed in Java. There was no attempt to study an object-oriented application development environment [6].

Approximately one-fourth of the semester was spent learning the fundamentals of Java applications; no Java applet programming was required or discussed in either course. In addition to the modules themselves, considerable time was spent on exceptions and I/O.

Important Aspects Of The Modules And Selected Comments From Students

In general, the effectiveness of placing the Java API documentation, the code itself, and the run from the code in close proximity was thought to be an effective teaching strategy, and indeed this turned out to be the case. Student comments illustrating this follow.

- Taking the program step-by-step, as we did last class, is to me one of the clearest and best ways I can learn Java. Thanks.
- API documentation was very useful in helping me to understand the many different ways a date can be retrieved
- The modules so far have been very helpful. You are doing a good thing by presenting small amounts of info each time. Easier to digest.
- I really enjoyed learning Java. The Java examples were wonderful. I would like some more modules to carry me through to the end.
- Even though this module seems straightforward, seeing it on paper, and its results, along with its GREAT comments helped me to understand it. Especially when the module asks Same Value? good stuff.
- These modules are great and I wish we had more of them....
- Amazing and interesting way to learn Java and refer to any problems we incur....Thanks!! :-)
- Anything is probably easier and more interesting than COBOL. Java is only easier than C because I know the syntax and logic of C programming and understand pointers well enough.

Computer Science vs. Information Systems Student Reactions To The JM

The following chart (and Figure A) shows that the following null hypothesis can be rejected at the .01 level for ST1, ST4, ST5, ST7, ST8, ST9, and ST10 (also see Figure A).

The following scale was used:

- 1.Strongly disagree
- 2.Disagree
- 3.No opinion/Not Applicable
- 4.Agree
- 5.Strongly agree

H₀: Computer Science and Information Systems students responded in the same fashion to the various individual statements for all 28 modules taken as a whole (2 tailed test)

H_A: Information Systems students responded differently (actually more positively) than Computer Science students. (L.B.=lower bound, U.B.=upper bound (both at .01))

Table 1. Responses to statements 1 through 6.

	ST1	ST2	ST3	ST4	ST5	ST6
Average	4.18	4.15	3.83	4.24	4.14	4.29
I.S.	4.24	4.14	3.84	4.34	4.24	4.29
C.S.	4.11	4.16	3.82	4.13	4.01	4.29
Difference	0.13	-0.01	0.03	0.21	0.24	0
Stdev	0.79	0.86	0.85	0.88	0.82	0.8
Stdev	0.82	0.79	0.84	0.94	0.87	0.73
L.B. (I.S.)	4.19	4.09	3.79	4.28	4.19	4.24
U.B. (C.S.)	4.17	4.2	3.88	4.2	4.07	4.35
reject at .01	Yes	No	No	Yes	Yes	No

Table 2. Responses to statement 7 through 10 and average.

	ST7	ST8	ST9	ST10	AVE
Average	4.02	4.19	3.48	3.63	4.01
I.S.	4.17	4.3	3.69	3.82	4.11
C.S.	3.84	4.05	3.22	3.39	3.9
Difference	0.33	0.25	0.47	0.44	0.21
Stdev	0.77	0.77	0.91	0.98	0.53
Stdev	0.97	0.8	0.89	0.82	0.47
L.B. (I.S.)	4.12	4.25	3.63	3.76	4.07
U.B. (C.S.)	3.91	4.11	3.28	3.45	3.94
reject at .01	Yes	Yes	Yes	Yes	Yes

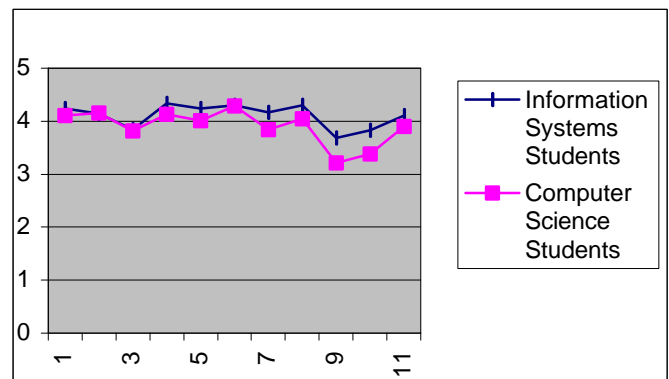
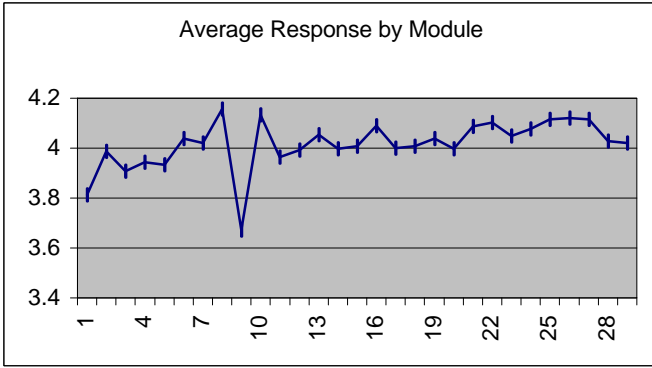


Figure A. A graph of the average responses to the individual statements for all 28 modules taken as a whole.

General Student Reactions to the JM

The following graph shows that overall student favorability to the module approach seemed to gradually increase over time, with the notable exception of the Reflection class (Module 9).



A graph of the overall average response by module for all students.

Figure B.

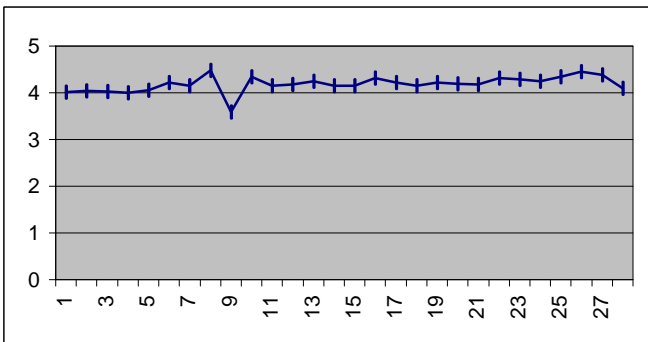


Figure 1. This module was very effective in helping me to learn Java.

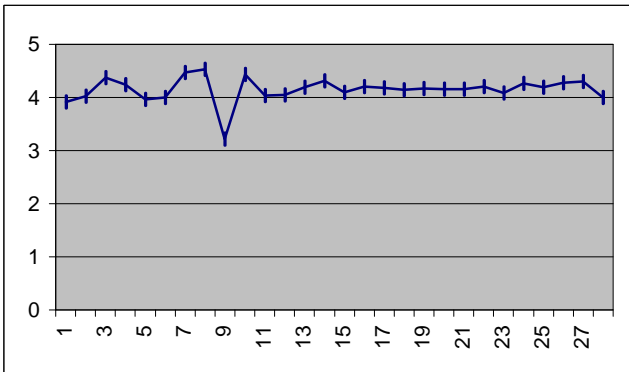


Figure 2. This module had a sufficient number of comments with the code.

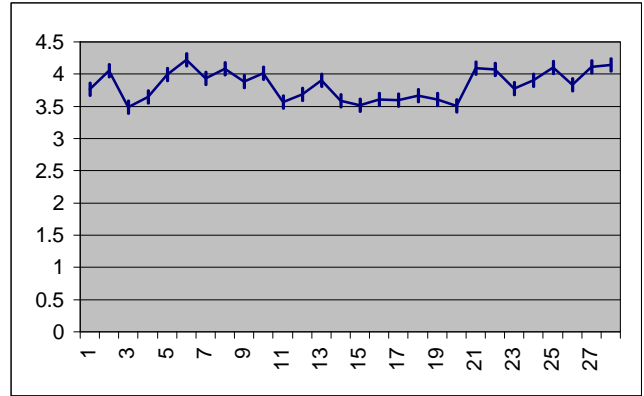


Figure 3. The Java API documentation was very helpful for this module.

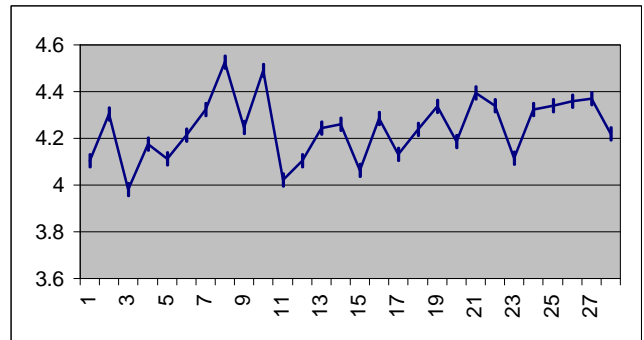


Figure 4. Seeing the run of the module was very important.

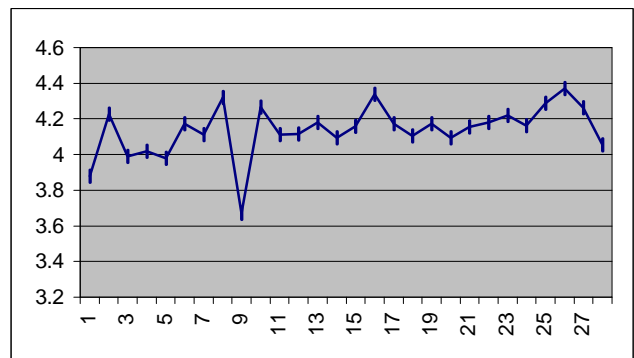


Figure 5. This module was constructed at an appropriate level for my background.

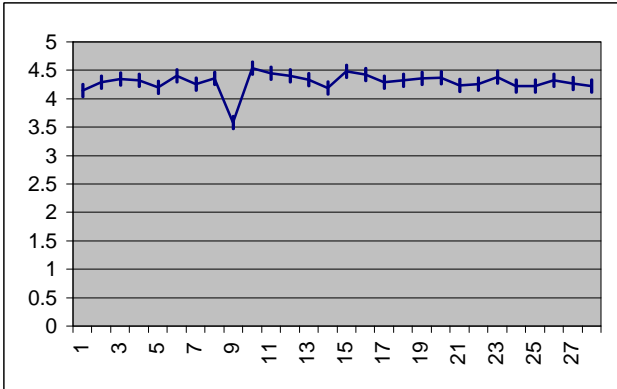


Figure 6. I believe that I could take this module and modify it for similar problems.

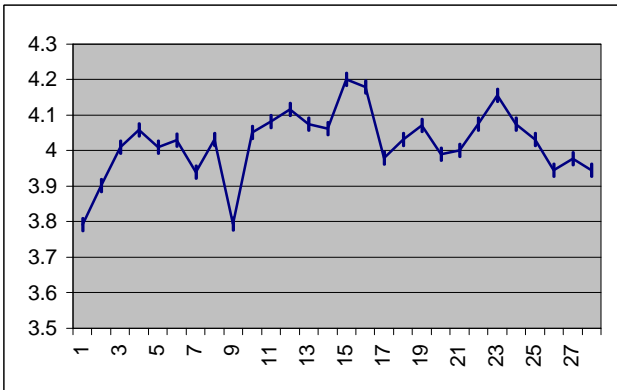


Figure 7. I believe that Java will be relatively straightforward to learn.

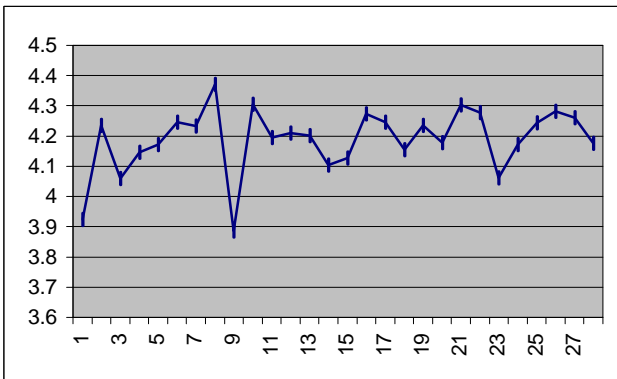


Figure 8. I believe that the amount of time I spent on this module was well worth it.

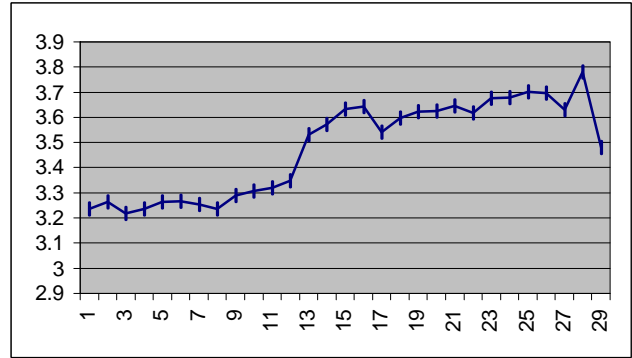


Figure 9. I believe that Java is easier to learn than C.

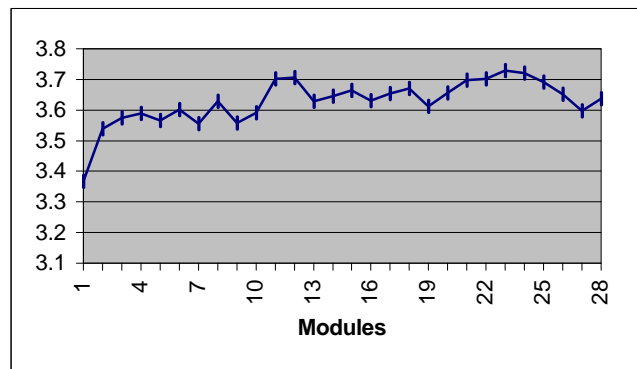


Figure 10. I believe that Java is easier to learn than Cobol.

Summary

The author believes that this approach can be successfully used in a variety of circumstances where the objective is to learn the fundamentals of object-oriented program quickly, so that students can use their newly acquired tools. Courses like CS1, IS1, and Data Structures do not have to be significantly modified from their procedural approach in order to give students a taste of the object-oriented paradigm. Students seemed to be very pleased with the courses as indicated below:

- 1) Information Systems students responded differently (actually more positively) than Computer Science students. (L.B.=lower bound, U.B.=upper bound (both at .01)) for each statement except
 - a. This module had a sufficient number of comments with the code;
 - b. The Java API documentation was very helpful for this module
 - c. I believe that I could take this module and modify it for similar problems.
- 2) Overall student favorability to the module approach seemed to gradually increase over time, with the notable exception of the Reflection class (Module 9).

- 3) The modules were very effective in helping to learn Java.
- 4) Almost all modules had a sufficient number of comments with the code
- 5) Students believed that the Java API documentation was very helpful for the modules.
- 6) The importance of seeing the run of the module varied considerably.
- 7) The level of the module for the student background had some variability.
- 8) Students generally believed they could take a module and modify it for similar problems (reusability).
- 9) Students believed that Java will be relatively straightforward to learn and their confidence generally improved over time.
- 10) There was some variability in the belief that the amount of time I spent on this module was well worth it.
- 11) There was a clear indication that students believed (more and more over time) that Java is easier to learn than Cobol.
- 12) There was a clear indication that students believed (more and more over time) that Java is easier to learn than C.

3) Koffman, Elliott and Wolz, Ursula, *CS1 Using Java Language Features Gently*, ACM SIGCSE Bulletin 31 (3), 40-43, 1999.

4) LaFore, Robert., *Data Structures and Algorithms in Java*. The Waite Group Press, Corte Madera, California, 1998.

5) Preiss, Bruno R. *Design Patterns for the Data Structures and Algorithms Course*, ACM SIGCSE Bulletin 31 (1), 95-99, 1999.

6) Warford, J. Stanley. *BlackBox: A New Object-Oriented Framework for CS1/CS2*, ACM SIGCSE Bulletin 31 (1), 271-275, 1999.

7) Weiss, Mark Allen, *Data Structures and Problem Solving Using Java*. Addison-Wesley Longman, Reading, Massachusetts, 1998.

Acknowledgement

The author gratefully acknowledges the work of R. Scott Runnion III who produced the scripting for the modules so that they could be submitted and analyzed electronically. In addition, grant support from Provost A. David Kline and the Office of Academic Affairs at UNF enabled the author to produce the modules.

References

1) Decker, R. and Hirshfield, S. *The Top 10 Reasons why Object-Oriented Programming Can't be Taught in CS!*, ACM SIGCSE Bulletin 25 (1), 51-55, 1994.

2) Goodrich, Michael T. and Tamassia, Roberto, *Using Randomization in the Teaching of Data Structures and Algorithms*, ACM SIGCSE Bulletin 31 (1), 53-57, 1999.