

Resources to Support the Use of Java in Introductory Computer Science

Eric Roberts
Stanford University
eroberts@cs.stanford.edu

SUMMARY

The growth in popularity of the object-oriented paradigm and the decision by the College Board to move the Advanced Placement Computer Science program to Java have led an increasing number of universities, colleges, and secondary schools to adopt Java as the programming language for their introductory computer science course. At the same time, many institutions find Java difficult to teach because of its significant detail complexity and its tendency to evolve rapidly over time. Such problems have led to periodic calls from within the computer science education community for the creation of simpler tools and a manageable Java subset specifically designed for presentation at the introductory level. Responding to this concern, the ACM Education Board has appointed a new task force whose mission is to develop an appropriate set of Java-based teaching resources that bears the endorsement of our professional society. The purpose of this special session is to introduce this task force to the SIGCSE community and to begin the discussions necessary to ensure that the deliverables produced by the task force will meet the needs of a wide range of institutions.

Categories and Subject Descriptors

K.3.2 [Computer and Information Science Education]: computer science education, curriculum.

General Terms

None.

Keywords

Computer science education, CS1, teaching libraries, Java.

1. INTRODUCTION

Since its introduction in 1995, Java has created quite a buzz in the computer science education community. If you look, for example, at the names of programming languages appearing in the titles of papers accepted for the SIGCSE annual symposium over the past eight years, references to Java outnumber those of all other programming languages combined. The trend toward the use of Java as the language for introductory computer science courses was evident as early as 1998 [8] and has since gathered additional momentum, bolstered in part by the decision of the College Board to move the Advanced Placement Computer Science program to Java in this academic year [1]. Some authors have suggested that Java has achieved a position of prominence within the academic community similar to that of Pascal in the 1980s, arguing that “we should shift our attention from the *whether Java* question to the *if Java, then how* question.” [10]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGCSE'04, March 3–7, 2004, Norfolk, Virginia, USA.
Copyright 2004 ACM 1-58113-798-2/04/0003...\$5.00.

Despite the fact that an increasing number of institutions are moving to adopt Java in their introductory curriculum, those institutions do not by any means report universal satisfaction with Java as a teaching language. If nothing else, the existence of paper titles such as “Java as first programming language: a critical evaluation” [5], “Java, the good, the bad and the ugly” [3], and “Java pitfalls for beginners” [2] offers evidence that there are problems associated with using Java at the introductory level. These problems are discussed in detail in a background paper in these proceedings [7], which argues that the difficulties observed in teaching Java are representative of a more general challenge facing computer science education as it tries to solve two self-reinforcing problems that have a negative effect on pedagogy:

- *Complexity*. The number of programming details that students must master has grown much faster than the corresponding number of high-level concepts.
- *Instability*. The languages, libraries, and tools on which introductory computer science education depends are changing more rapidly than they have in the past.

The issue of complexity, as it arises in Java and any full-scale modern language, has been widely recognized as a critical problem for introductory courses. In his keynote address at ITiCSE 2002, Niklaus Wirth railed against the complexity of modern languages, specifically including Java, lamenting that “People seem to misinterpret complexity as sophistication.” [11] He further argued that the responsibility for correcting the problem must lie with educators, and not with those commercial industries that have created the overly complex systems.

The background paper [7] continues this line of reasoning by proposing that any Java standard that is both stable and simple must come from the computer science education community. To this end, the ACM Education Board has initiated a new project with the following general charter:

To review the Java language, APIs, and tools from the perspective of introductory computing education and to develop a stable collection of pedagogical resources that will make it easier to teach Java to first-year computing students without having those students overwhelmed by its complexity.

The deliverables for the project include

1. *A definition of a subset of the standard Java APIs appropriate for first-year computer science*. This subset would involve restricting both the number of classes used as well as the number of public methods made visible within those classes. Note that this subset must be sufficient to have students write significant applications using Java. To this end, it will presumably be a superset of the AP Java subset [3], which seeks to define what aspects of the language will be tested on the AP exam.
2. *A public web site containing an updated JavaDoc reference manual for the approved Java subset*. This web site would make it possible for students to browse the standard classes and methods defined in the subset without being overwhelmed by classes, methods, and concepts they are unlikely to use. For the classes and methods that are included, the web site will contain more examples and tutorial material than is currently supplied with the Java APIs.

Figure 1. Task force members

Eric Roberts, Stanford University (chair)
Kim Bruce, Williams College
Scott Grissom, Grand Valley State University
Karl J. Klee, Alfred State College
Susan Rodger, Duke University
Fran Trees, Drew University
Ian Utting, University of Kent, Canterbury

3. A collection of pedagogically oriented APIs that have been evaluated and approved by the task force. The goal of the task force is not to privilege one particular set of extensions but rather to establish a process by which extended libraries that will be useful in teaching the introductory courses can be reviewed and standardized. Ideally, the review process will ensure that all approved packages had a common documentation scheme and compatible structures. The other major purpose of the review process would be to provide stability in the library packages by imposing a conservative release policy so as to protect users from having to adjust to frequent incompatible releases.

4. A survey and evaluation of existing noncommercial materials and tools for teaching Java. These tools will presumably include a range of resources including programming environments, "nifty" assignments, algorithm animations, and potentially much more. These resources can then be made available through the CITIDEL component of the National Science Digital Library (NSDL).

5. A proposal for sustaining the activity begun by this task force. Because computing is a rapidly changing field, it is foolish to expect the resources produced by the original task force to survive unchanged over the long term. The best that one could hope for is a system that provided some increase in stability along with a well-designed mechanism that permits evolution without putting too much strain on educational institutions. Developing such a strategy will presumably require the task force to identify potential revenue streams to support ongoing activity.

The work of the project will be conducted by a new task force chartered by the ACM Education Board, with the membership shown in Figure 1. The members of the task force have been chosen in part for their experience with introductory Java education but also with the goal of representing a broad range of constituencies. The task force includes instructors of first-year courses at a variety of institutions, including research universities, liberal-arts colleges, two-year colleges, and a non-US university. It must also include members who have considerable experience with the AP Computer Science program and who understand the special problems of teaching Java at the precollege level.

The proposed timetable for the project includes the milestones shown in Figure 2. These resources will be made available on the web and distributed through such channels as CITIDEL and JERIC.

The success of the project will depend in large measure on its ability to secure participation and support from the broader computer science education community beyond the small number of people on the task force itself. To this end, the public presentations at the SIGCSE symposium in 2004 and 2005 are essential. The symposium session in the first year allows us to introduce the project and to issue a call for participation, primarily in the form of a request that members in the community submit their own materials for review. The session in 2005 gives the project a chance to present its preliminary results at a time that still allows for revision before the target completion date in the summer of 2005.

Figure 2. Proposed project milestones

Jan 04	Convene initial meeting of full task force
Mar 04	Introduce the project at a SIGCSE special session Call for submissions of proposed pedagogical APIs
Apr-Nov 04	Conduct series of working meetings Undertake review of existing resources Review submissions of pedagogical APIs Refine definition of Java teaching subset
Dec 04	Publish draft of Java teaching subset on the web
Mar 05	Present complete draft report at SIGCSE 2005
Mar-May 05	Review and incorporate feedback from SIGCSE Construct website
Jun 05	Publish final report and website in time for fall

This project has been endorsed and given partial funding by the ACM Education Board and the SIGCSE Board. At the time of this writing, we are applying for additional support and hope to report that the project has been fully funded by the time of the SIGCSE symposium.

REFERENCES

1. Owen Astrachan, Robert (Corky) Cartwright, Gail Chapman, David Gries, Cay Horstmann, Richard Kick, Frances Trees, Henry Walker, and Ursula Wolz. Recommendations of the AP Computer Science ad hoc committee, October 2000.
2. Robert Biddle and Ewan Tempero. Java pitfalls for beginners. SIGCSE Bulletin, 30:2, June 1998
3. Peter Martin. Java, the good, the bad and the ugly. SIGPLAN Notices, April 1998.
4. The College Board. Advanced Placement Program Course Description: Computer Science. New York: The College Board, May 2003. http://www.collegeboard.com/ap/pdf/cd_computer_science_03.pdf.
5. Said Hadjerrouit. Java as first programming language: a critical evaluation. SIGCSE Bulletin, June 1998.
6. Eric Roberts and Gerald Engel (editors). *Computing Curricula 2001: Final Report of the Joint ACM/IEEE-CS Task Force on Computer Science Education*. Los Alamitos, CA: IEEE Computer Society Press, December 2001. <http://www.acm.org/sigcse/cc2001/>.
7. Eric Roberts. The dream of a common language: The search for simplicity and stability in computer science education. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, VA, March 2004.
8. Chris Stevenson and Tom West. Language Choice and Key Concepts in Introductory Computer Science Courses. Journal of Research on Computing in Education, Fall 1998.
9. Chris Stevenson. Java engagement for teacher training: Proposal for a pilot project to help local secondary computer science teachers develop expertise in Java programming. ACM memorandum, August 2002.
10. Chris Wallace, Peter Martin, and Bob Lang. Not whether Java but how Java. Paper presented at the Java in the Computing Curriculum conference, London, January 1997.
11. Niklaus Wirth. Computing science education: The road not taken. Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education, Aarhus, Denmark, June 2002.