

Input/Output for a CS1 Course in Java

Some Considerations

Elliot B. Koffman

Computer and Information Science Department, Temple University, Philadelphia, PA, USA
koffman@temple.edu

Abstract: This paper discusses considerations for input/output for a first course in Java. It includes the opinions of several computer science educators regarding the pros and cons of using a non-standard package for input/output and the requirements of such a package. It also describes the evolution of a simple package for input that it is easy to implement using the Swing class and gives guidelines for using standard Java for simple input/output.

Key words: Computer Science 1 (CS1), Novice programming in Java

1. INTRODUCTION

1.1 Non-standard packages for input/output

Many Java textbooks for CS1 use simple packages for Java input/output. Some classes that were developed to simplify console input/output are: `ConsoleReader` [3], `Text` [2], and `SimpleInput` [1]. These classes were written because the textbook authors and teachers of CS1 recognized that console-based input in Java is difficult for novices to use. At the same time, a few packages were also written that permitted GUI-like interactivity. Examples of these packages and classes are: `simpleIO` [7] and `Java Power Tools` [8]. These packages were developed because their implementers recognized that console-based input/output was tedious and uninteresting to students who were accustomed to applications with Graphical User Interfaces (GUIs). The package developers also felt that novice students should be able to write programs with GUI-like interactivity before

mastering the Java APIs that would enable them to build their own GUIs: AWT and Swing.

1.1.1 Educator comments on using non-standard packages

In early April, 1999 there was a thread on the SIGCSE listserve that discussed the use of non-standard packages for teaching CS1 in Java. The messages seemed to be split fairly evenly between those who advocated their use and those who were against their use. The following comment by H. Conrad Cunningham (University of Mississippi) succinctly summarizes the two points of view:

"On one hand, use of simplified I/O or GUI packages can make Java easier to teach and use in CS1/CS2. So the various packages that are available with various textbooks can be helpful. And we want to convey to our students that it is easy to add on significant capabilities to Java and other such languages.

On the other hand, there is concern that use of a non-standard, add-on package might be confusing to first year students. Students tend to take what they learn as being part of Java. When they move to another environment, they are somewhat lost and confused. They have the overhead of relearning an I/O package or installing the one they are accustomed to. There is thus some sentiment to only teaching the "standard" (whatever that means with Java)."

A. Joseph Turner (Clemson University), makes the following point in favor of using packages:

"I haven't tried teaching CS1 or CS2 using only the classes in the standard Java library, but unless you want to limit yourself to reading only a character at a time or a line at a time (as a string), you don't want to use only the I/O facilities in standard Java. (There are some ways to do a bit better than reading only characters or lines, but the overhead and complexity of doing so are out of the question for beginning students as far as I am concerned.)"

David Arnow (Brooklyn College of City University of New York) makes the following argument against the use of packages:

"There is another argument in favor of NOT providing a special I/O package for student use in CS1/Java. While constructions such as

```
BufferedReader br = new BufferedReader(
    new InputStreamReader(
        new FileInputStream(
            new File("myinput"))));
```

may be introduced as "boilerplate", they also provide an opportunity to:

- emphasize the idea of classes as models, as repositories of particular sets of behavior.
- illustrate the use of composition
- foster the notion and UTILITY of abstraction

In connection with the latter, students, even CS1 students, can appreciate the fact (and the accompanying discussion) that the last two lines of the above code can be replaced with

```
System.in
```

or

```
new URL("http://www.acm.org").openStream());
```

or

all sorts of other things eventually

because each of these things play the "role" of an `InputStream`. In other words, if one's goal is to take an object-oriented approach in CS1 Java's I/O classes can be viewed as an asset, not a liability. (As a side note, this i/o stuff usually turns out to be the LEAST of the students' problems in CS1.)"

Michael G. Branton (Stetson University) advocates teaching basics of the AWT first instead of using packages:

"You can also do it with Java without that much fuss. I've taught the course 3 times now using AWT. you don't need very much of it to put a text field or 2 and a button on the screen, folks. There seems to me a lot of hand-wringing over this, but I've found that my students don't find this difficult to deal with at all. It's the algorithm development that's still the "hard part." My experience is that teaching them enough of the GUI for CS1 doesn't take much time at all. I still have plenty of room for the topics I consider important, and text fields and buttons give them something very concrete (in a virtual sort of way :-)) to think of as objects right off the bat."

Finally, Judy Bishop (University of Pretoria) makes the following points in favour of non-standard packages:

- "Java is an extensible language, with extensions coming via packages (APIs). In today's world, the ability to harness this power is more integral and more important for fresh minds to understand in *week one*, than a long-winded object instantiation, four brackets deep.
- An I/O class is an excellent case study in its own right as it illustrates exception handling, as well as tokenizers and string functions. It also does not have to be long: our `Text` class is 2.5 book pages of code, providing 10 methods for input, output and file opening.
- If a lecturer feels very strongly about teaching Java at the rock face, and is not convinced by point 1, then I would still suggest that when writing your own package! comes along in the course (which it ought to at some stage), then an I/O or Graph package is an ideal candidate."

What have we learned from this thread?

Informatics Curricula and Teaching Methods

IFIP TC3 / WG3.2 Conference on Informatics Curricula,
Teaching Methods and Best Practice (ICTEM 2002) July
10-12, 2002, Florianópolis, SC, Brazil

Cassel, L.; Reis, R. (Eds.)

2003, XVI, 152 p., Hardcover

ISBN: 978-1-4020-7266-6