

# Re: Interesting article by Joel Spolsky: The Perils of JavaSchools

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.programming/2006-01/msg00221.html>

---

- *From:* "Arthur J. O'Dwyer" <[ajo@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:ajo@xxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 10 Jan 2006 13:42:07 -0500 (EST)
- 

On Tue, 10 Jan 2006, Gerry Quinn wrote:

johnnyb@xxxxxxxxxxx says...

[Gerry Quinn, IIRC, wrote:]

And it's the same half-baked point that has been made for millennia, when those who consider themselves as members of an intellectual elite get worried that their inferiors are achieving much the same things as they can, without having bothered with the elaborate membership initiations. "They must be stopped", they wail. "How can the sacred work be left in the hands of ignorami?".

Not at all. That is, I think you've hit the nail on the head with respect to the elite's attitude, but somehow you've come to the bizarre conclusion that "inferior" programmers are indeed achieving "much the same things" as the good, smart programmers. Would you disagree that most of the software produced today is extremely poor in quality? I can't blame Spolsky for making a connection between

- A) many college graduates don't display knowledge of basic software concepts
- and
- B) the software produced by many college graduates is low-quality

He's saying --- and I agree --- that if you teach students to think about software in the right way, it will make it easier for them to write better software later in life.[1]

If you don't understand tail recursion, how will you know that you need it, or, having it, know that it changes the question of whether or not to use a recursive procedure?

## Re: Interesting article by Joel Spolsky: The Perils of JavaSchools

That's a bit like the 'proof' question. What you need to know is what it delivers in terms of time and space efficiency. Then if for some problem you wanted to use recursion rather than iteration, you could evaluate whether that was feasible.

Here I agree with Gerry. The whole "tail recursion" thing is a complete and utter red herring in the JavaSchools debate. I didn't know Java had partial support for TRE, and I still don't care. I don't even care that Lisp supports TRE. That's an implementation detail, and it doesn't affect the way students /think/ about programming.

What matters is that Lisp (Scheme, ML) teaches students to use recursion, because they have to. Java does NOT teach students to use recursion, because they don't have to. See below.

When they need to learn it, they will. Hopefully employers will choose to hire staff capable of doing or learning to do what they need, whether that includes the above things or not.

(Silly. Spolsky's point is that if an interviewer is faced with a JavaSchool grad, he won't be able to judge the interviewee's capability, precisely because the interviewee won't be able to demonstrate any grasp of the concepts it take capability to understand. If Alice understands pointers and recursion, she'll be able to understand anything, and get hired. If Bob doesn't understand P&R, he may yet be able to understand anything, but he won't be able to demonstrate that capacity, and he won't get hired.)

Which is exactly why Java is a bad language to use for teaching. It doesn't teach them the skills they need to handle new situations.

Why not? Certainly there's no difficulty using recursion functions in Java.

Certainly. Which means that any highly motivated student will go far. Duh. We already knew that. Spolsky's trying to make sure that (A) the completely unmotivated students /don't/ go far, but switch out of a major that's not their cup of tea; and more importantly, (B) the somewhat motivated slackers who compose the vast majority of [U.S.] college students /do/ go far, by being forced to learn pointers and recursion and thereby expand their horizons beyond Visual Basic.

This is a point you (Gerry) have been stubbornly ignoring through this whole thread --- students are lazy. If they can write Visual Basic code in Java, they will. This is how so many students come to the

## Re: Interesting article by Joel Spolsky: The Perils of JavaSchools

200-level courses, even at CMU, unable to write 'atoi'.  
Lisp, Scheme, and ML make it impossible to cruise through the curriculum in Visual Basic mode. They force the students to learn, which is all that most students need.

[In re: another red herring]

Ironically, the original argument was that by not supporting iteration - in other words because of their limited expressiveness - certain languages make programmers better.

Right. That's correct. See above.

Yet by not supporting pointers, Java is supposed to make them worse.

I have two comments on the above sentence, which I realize don't take into account the thread of the debate up to this point: (A) Java /does/ have pointers, it just hides the \*s; and (B) the reason that Java is not a replacement for Lisp/Scheme/ML in universities has nothing to do with pointers. (Lisp/Scheme/ML don't have pointers, either![2])

And its support of recursion doesn't count, for some reason.

Right --- because /supporting/ recursion is a far cry from /encouraging/ recursion. The average student will never learn to fish by being put out on a lake that /supports/ fishing; they need firm guidance. (And no, you can't entirely blame the teachers. A lot of teachers are bad, but it's better that a bad teacher teach Lisp than a bad teacher teach Java, and there appears to be a shortage of good teachers at the moment.)

-Arthur

[1] - Also consider that Spolsky, as one of the "elite," isn't necessarily thinking of future application-software drones. His argument applies even more forcefully to future visionaries --- the ones who are going to make your life easier with totally original paradigms, whether it be a new language, a new operating system, a new compression or network protocol, or a new kind of computer altogether.

[2] - Okay, ML at least supports pointers, but they're ugly as sin and come with severe limitations --- no pointing to objects "on the stack",

Re: Interesting article by Joel Spolsky: The Perils of JavaSchools

for example. I bet Lisp has an even uglier pointer syntax; I wouldn't be surprised if Scheme didn't have pointers at all, and I would be surprised if either language allowed pointing to stack objects.

Point is, in Java the teacher does pointers on day 3 --- you need pointers and "new" in order to deal with any object more complicated than String. In a functional language, pointers are a hackish afterthought

that will likely never be mentioned in class. You'll get enough pointers in 15-113 and 213. :)

.