Some Remarks on Teaching of Programming

Janusz Jabłonowski

Abstract: *In this paper we give some remarks on the problem of teaching programming for the first year students at the undergraduate level. The remarks are based on the experience gained by the author through several years of teaching programming at the Warsaw University. The main issue discussed in this paper concerns the choice of the programming language for the first course in programming. Earlier the language of choice in many places around the world was Pascal or C. Then some universities decided to switch to C++. Nowadays many introductory courses are taught in Java or C#. We try in this paper to advocate the choice of Pascal.*

Key words: *Programming Languages, Teaching of Programming, Pascal, Java, Smalltalk, C#, C++, C.*

INTRODUCTION

In this paper we give some remarks on the problem of teaching programming for the first year students at the undergraduate level. The remarks are based on the experience gained by the author through several years of teaching programming at the Warsaw University. The main issue discussed in this paper concerns the choice of the programming language for the first course in programming. Earlier the language of choice in many places around the world was Pascal or C. Then some universities decided to switch to C++. Nowadays many introductory courses are taught in Java or C#. We try in this paper to advocate the choice of Pascal. Of course in any concrete situation there are many issues to be taken into account, which may force other choice. Among these topics are:

- the background of the students,
- the amount of other programming, or closely related to programming, courses in the curriculum,
- the profile of the university,
- the expected expertise of the graduate.

So, after making precise to what we are comparing teaching in Pascal, we start with describing our assumptions of the study model. Then we discuss the virtues and drawbacks of starting programming teaching with Java. Then we conclude with some final remarks.

TO WHAT WE COMPARE?

We have chosen Java as the language of comparison. Why we concentrate on that language? In our opinion Java is a good representative of the (quite) new languages, which are commonly considered as the first teaching programming language. Virtually all comments on Java in this paper can be equally well applied for example to C#. We do not criticize C [6] or C++ [7] as the first programming language, not because we consider them as more appropriate for this task. On the contrary, in our opinion they are much worse for this job. C was never meant by its designers to be used for teaching, neither did they consider readability and clarity of the program code or safety as the main issues of the development. C++ inherits many bad features from C, but the main reason against C++ as the introductory programming language is that it is just to difficult - its semantics is very sophisticated and it requires a long time to start to program in a real C++ (and not in a quite small subset of the language, with no knowledge of many issues happening in the program under the cover). Due to the lack of space we do not go into details on this topic.

WHAT WE DO CRITISICE AND WHAT NOT?

We are not criticizing Java (nor C++ or C) as a language for programming. In fact the author is teaching programming in all of these languages, and is using all of them on a regular basis. We think that a computer scientist or software engineer should know all of them, because each one of them is interesting and has its own natural area of application. That what we are criticizing is the use of one of these languages as the first programming language for the introductory course on the undergraduate level.

THE STUDY MODEL UNDER CONSIDERATION

As we have already mentioned there are many issues which may influence the choice of the first programming language. In this section we try to summarize the adopted assumptions.

The first one is of course concerned with the kind of study. We assume an undergraduate study at a university or academia. More we assume, that this is a study in the area of software engineering or computer science. This is not vital for our discussion, but we also assume a three years period of study on the undergraduate level (as it is in most European countries).

We think that it is normal, and even more: it is crucial for students on such kind of study, to learn more than just one programming language. Why is it important to spend time on discovering subtleties of various syntaxes and semantics instead of teaching just one carefully selected language? There are many reasons:

1. There is no best programming language - so whichever we would choose, we will always be criticized by our students, that we are not teaching them the best programming language.
2. There is no dominant programming language (thanks to heaven, by the way). So our students cannot be assured, that they will with high probability work with the one chosen by us.
3. Languages are changing (the languages itself and the collection of the so called "modern" languages), so we cannot assure our students, that the chosen by us language will still be considered modern after the study. And (what is most important), the students during their professional life will be forced to switch to other language, what is quite easy for somebody who knows several programming languages and is extremely frustrating and difficult for somebody, who knows only one.
4. There is no "all purpose" language. Many tasks require specialized languages (like for example HTML to describe WWW pages, XML to describe textual tagged data, many script languages to describe dynamics of WWW pages, SQL to describe database queries, C for system programming under Unix, C++ for writing time-critical OO-systems, etc.). In many situations we can choose one language from many (anyway general purpose programming languages are aimed at being applicable in many areas of application), but nevertheless even then, some from the possible languages fit better to the task at hand than the others. The ability to choose the proper tool for given task is very important and surely requires some practice and of course some knowledge of the possible choices.
5. When one starts to learn the first programming language, one tends to consider syntax of this language as something of deep importance. Only after learning the second, third language one realizes that syntax is of very limited significance - we must know it to write syntactically correct programs, nothing

more. There is quite similar situation with language semantics and idioms specific to that language. If one knows only one language it is hard to see, that for example the notion of a variable is by far more important than the way in which variables in a particular language are being initialized (or whether they are initialized automatically or not). Knowing the same notions in various programming languages lets see them from various viewpoints and therefore enables deeper understanding of these notions.

Hence we assume that the curriculum includes many courses during which new programming languages are being taught (it does not mean of course that the entire course is devoted to teaching only the language - we rather expect courses in various domains, i.e. object-oriented programming or concurrent programming or Internet programming, to contain presentation of appropriate tools, among them programming languages).

SOME COMMENTS ON ARGUMENTS FOR CHOOSING JAVA AS THE FIRST PROGRAMMING LANGUAGE

As we mentioned before, we treat Java only as a (good) representative of the languages, which are commonly considered as a candidate for the language of choice for the introductory programming course. In this section we go step by step through the arguments which are commonly used to advocate the choice of Java and give some (critical) comments to them.

Java is popular (in the sense: commonly used)

It is sad to notice that nowadays the popularity is to much degree connected with the funds given by the owner of the language to its promotion (probably C++ was the last programming language, which is commonly used around the world and has not been substantially financed by its owner). It means, that the popularity may easy change just with the changes on the finance markets. Therefore it does not seem reasonable to connect the future and success of our students with any particular market product. Let us stress again that we do not say that Java is not worth teaching - certainly is! But not necessary as the only (most popular or dominant) language. In our opinion it is better to teach it to more matured students, who can better appreciate its virtues.

Java is modern

The problem with being modern is such, that things considered modern at the time of starting study do not have to remain modern after 3-years study period and certainly will not be modern after ten or twenty years (that is when students will work using the knowledge gained during the study). So we do not think that the aspect of being modern is to be considered when choosing the first programming language.

Java has strong typing

This is a very important virtue. Languages with strong typing are generally safer and more adequate for teaching of programming. Fortunately we have many such languages to choose from (e.g. Pascal, C#). So this only virtue is not differentiating enough to guide us in the selection of the proper language. Java has unfortunately some mechanisms which allow to break the type safety more or less knowingly (with explicit or implicit type-casts, the good news is that the new language version from J2SDK 1.5 will require from programmers less type-casting).

Everyone starts teaching with Java

Quite a poor argument, one can hear that some years ago everyone was starting with C (fortunately it is not true), and that later everyone was starting with one of functional programming languages. It is true that many universities use Java as the first language, but this can be said about quite a few other languages.

Java compiler has good diagnostics

Unfortunately diagnostics generated by Java compiler (and by all compilers of large programming languages) are not friendly for starting users. It is not the drawback of the compiler itself, it is just the complexity of the language - if there is an error, there are so many possible reasons for it, that just displaying some of the most probably may confuse not only the beginner. Only a compiler of a simple language (like genuine Pascal, but not for example Object Pascal), is in a so comfortable situation, that it is *possible* in most cases to clearly state to the user the cause of his or her mistake. And of course only good compiler will take advantage of this possibility.

Java may be the only language taught during entire study

That is true. It might be the only language due to its flexibility - it may be used equally well during the course in object-oriented programming as in a course in concurrent programming. But the question is: do we need only one programming language for the entire study? There are circumstances in which the answer will be yes (for example when during the study there is only one or two courses related to programming), but in the case of software engineering or computer science studies the answer is certainly negative, as we have pointed out in the previous section.

SOME CRITICAL REMARKS ABOUT JAVA AS THE FIRST PROGRAMMING LANGUAGE

Now we will summarize our critical remarks about starting teaching programming with Java.

It is simply too big

It is not possible for a first (and not only first) year student to learn the entire tool. We must remember that in Java not only the language itself is important but also the entire environment around it, i.e. many very interesting, very useful and very rich libraries. Unfortunately they are also very large. And without knowing them, students will still have the feeling that they just started learning something very big. This feeling at the beginning of a study is completely understandable, but at the end of the first year is very frustrating.
But it is not only the problem of the libraries, it is the problem of the language itself, which is getting more complicated with each version. Lets us compare: the specification of Java [4] (without libraries, description of the new language version surely will be larger) has more than 500 pages, whereas Pascal standard [5] has only less than 90 pages.

Has many mementos taken from C

C's syntax never was said to be especially elegant. Unfortunately many awkward C constructs somehow survive till Java times. Let us mention just few of them. The ugliest

form of selecting statement (called switch). The unnatural way of denoting equality (== instead of standard =) and the equal sign used to denote assignment.

Has many traps in its syntax and semantics

Let us point out just some of them. These are valid Java constructs, which mean something opposite to the intuitions of most persons:

- The if condition in the following program is presumably false, although both strings are the same:
  String S = "A";
  if ("AB" == S + "B")
  The reason is the lack of equality operator for strings. In the above example we have tested identity of objects instead equality of object values.
- The following condition is false, although both equality sides seem to have the value 10:
  if (10 == 010)
  The reason is, that the literal 010 means a number in octal notation.
- The following condition is false, although both sides seem to be equal to 1.5:
  if (3/2 == 3/2.0)
  The reason is that the / operator applied to two integer arguments means integer division (yielding 1 here) but if one of its arguments is a real number, then also the division is a real number division (here yielding the result 1.5).

Due to the lack of space we omit here rest of our examples. We sum up with a comparison of the first program usually shown during a course on programming in a new language (the "Hello world" example) written in three languages: Java, Pascal and Smalltalk.

The Java version:

```
public class HelloWorld
 {
  public static void main(String[] args)
   {
        System.out.println("HelloWorld!");
   }
 }
```

The Pascal version:

```
program HeloWorld(input, output);
  begin
    writeln('Hello World!')
  end.
```

And finally the Smalltalk version:

```
Transcript show: 'Hello World!'
```

Smalltalk has the shortest and Java the longest version. But it is not the size of code which is most important. By far it is more significant how many notions in each language are needed to understand the given example. Due to the lack of space we do not list here

precisely all the notions used in each example, but again Java is here the most demanding one.

CONCLUSIONS AND FUTURE WORK

The problem of selecting the first programming language used for teaching introductory programming is quite complicated and depends on many issues, also those closely related to the specifics of the particular university. In this paper we tried to show that one quite reasonable solution is still using Pascal as the first programming language, mainly because of its simplicity. It is definitely not the only proper choice, but it is one with many virtues. It does not mean that students will not learn object orientation (or functional programming or programming in logic), it is just a selection of (in author's opinion) proper tool for the task of an introductory course. Also classical languages and environments, like for example those of Smalltalk, can be used for this task with success.

It is worth to add that there are other approaches to this problem, for example by somehow simplifying the Java language and giving proper environment for it, as is done in the BlueJ project [2] or by supporting the learning process with proper tutorials as for example one for C# [1].

REFERENCES

[1] Bishop,J., Horspool,N.; C# Concisely; Pearson 2004.
[2] Fisker,K., Koelling,M.; The BlueJ Environment, Reference Manual, 1.1; 2004; http://www.bluej.org/reference/BlueJ-reference.pdf.
[3] Goldberg,A., Robson,D.; Smalltalk 80: The Language (The Blue Book); Addison-Wesley Pub Co; 1st edition; 1989.
[4] Gosling,J., Joy,B., Steele, G., Bracha, G.; The Java Language Specification, second edition, 2000; Sun Microsystems.
[5] ISO; Pascal ISO 7185:1990.
[6] Kernighan,B.W., Ritchie,D.; C Programming Language (2nd Edition); Prentice Hall PTR; 2nd edition; 1988.
[7] Stroustrup,B.; The C++ Programming Language (3rd Edition); Addison-Wesley Pub Co; 3rd edition; 1997.

ABOUT THE AUTHOR

Senior lecturer Janusz Jabłonowski, PhD, Department of Mathematics, Informatics and Mechanics, Warsaw University, Phone: +48-22 5544401 E-mail: janusz@mimuw.edu.pl