Teaching Java with the BlueJ Environment

Dianne Hagan
School of Computer Science and Software Engineering
Monash University, AUSTRALIA
dianne.hagan@infotech.monash.edu.au

Selby Markham
Faculty of Information Technology
Monash University, AUSTRALIA
selby.markham@infotech.monash.edu.au

Abstract

BlueJ is a visual programming environment designed to teach object-oriented programming, using Java as the implementation language. BlueJ allows students to concentrate on solving programming problems without becoming distracted by the mechanics of compiling and executing Java programs. This paper reports on the first use of BlueJ to teach Java to an introductory programming class, in a computing degree in 1999. Several mechanisms were put in place to help students with any problems they encountered. Surveys and interviews were used to collect data on student backgrounds, perceptions and attitudes towards BlueJ. In spite of some problems encountered with installing and running the software, students who participated in the study generally found that BlueJ was helpful in learning Java.

Keywords

Introductory programming environment, BlueJ, Teaching Java, Object-oriented programming

Introduction

Many students find programming a difficult thing to learn. As noted in (Soloway, 1986), the problem is often that students have difficulty "putting the pieces together" – they tend to focus on the syntax and mechanics of a programming language and have difficulty seeing the "big picture" of what they are trying to do. Visual programming environments attempt to overcome this problem by providing graphical representations of this big picture. Systems such as Alice (Dann, Cooper & Pausch, 2000) allow students to write programs using an interactive visual environment, which gives them a good understanding of some programming concepts, but removes the necessity to grapple with others. It is unclear how well these students do when they attempt to translate these concepts into a "real" programming language.   Vis-Mod (Jimenez-Peris, Patino-Martinez & Parcios-Martinez, 1999) is an example of another kind of system that visualizes the execution of a running Modula-2 program, thus helping the student to understand and debug it, but does not provide much help in writing the program in the first place. The amount of information it displays in several windows, while useful to an experienced programmer, could be overwhelming to a novice. Our experience is that students are reluctant to learn a programming language that is not commercially relevant.

BlueJ is an integrated development environment, based on the language Blue which was designed and implemented by Michael Kölling and John Rosenberg (Kölling & Rosenberg, 1996) to teach object-oriented programming.   BlueJ has the same visual environment as Blue, but uses Java (Gosling, Joy & Steele, 1996) as the implementation language. Java is a language that is currently popular in industry, and students beginning the course are highly motivated to learn it. In 1999, BlueJ was used for the first time to teach Java to introductory programming students. We had a class of about 350 students in the first year of a computing degree. A number of these students had some prior knowledge of programming, but most did not.   The students were told that they were using a Beta version of BlueJ which was still under development, but that the teaching staff believed that it was nevertheless an improvement upon alternative programming environments. Students were warned to expect some bugs in the software, and were promised that, if they reported their difficulties promptly, every effort would be made to resolve them as soon as possible. A website was created to enable them to report their difficulties with BlueJ and to read the staff responses to problems encountered by other students. Most students were pleased at the idea of being the first to use this new environment and contributing to its development and improvement.

Advantages of the BlueJ Environment

BlueJ is designed to facilitate learning of object-oriented concepts. Its visual environment makes it possible for novice students to interact with existing classes without writing any code at all, for the first few weeks of their programming course, until they understand the basic concepts of object-oriented programming. However, it is necessary for them to learn how to write Java code by the end of the semester. BlueJ's editor (see Figure 2) helps them to do this, and to debug their programs. In BlueJ, objects can be instantiated from classes and inspected to ascertain their state (see Figure 4), and requested to execute their public methods. Returned values from methods are shown in a window, and output to the screen in another window. Students are prompted for values of method arguments when appropriate.

Classes are depicted as rectangles in the main window, and objects are shown as ovals in the object window at the bottom of the screen (see Figure 1). A class has a dark border around it if it has compiled successfully, and is striped otherwise (see Figure 3). The student clicks on a class to select it, then a right mouse click brings up a menu of all the public methods of that class. The student can select a constructor to create an object, and is given the opportunity to name the new object. If it is a constructor that requires arguments for the values of the attributes of the new object, the student is prompted to key in those values. The new object is then shown in the bottom window, labeled with its name. It is not necessary for students to write a `main` method for a program, as they can ask an object or a class directly to execute any of its public methods. However, it is possible to include a `main` method in a program, if that is desired. When classes are brought together from various directories into the BlueJ environment to construct a program, BlueJ automatically creates a package, and students rarely have to use the `import` statement in their programs. The BlueJ environment makes testing a class easy, as students can invoke each method in the class directly and see the output and returned value. They can create a method to test all the other methods systematically, and invoke it directly. They can also include such a method in a test class whose purpose is to test another class thoroughly.

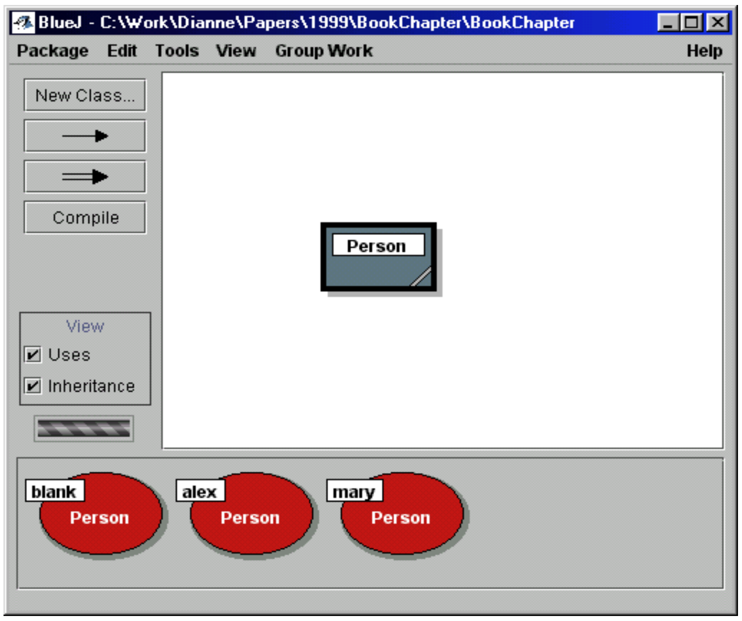
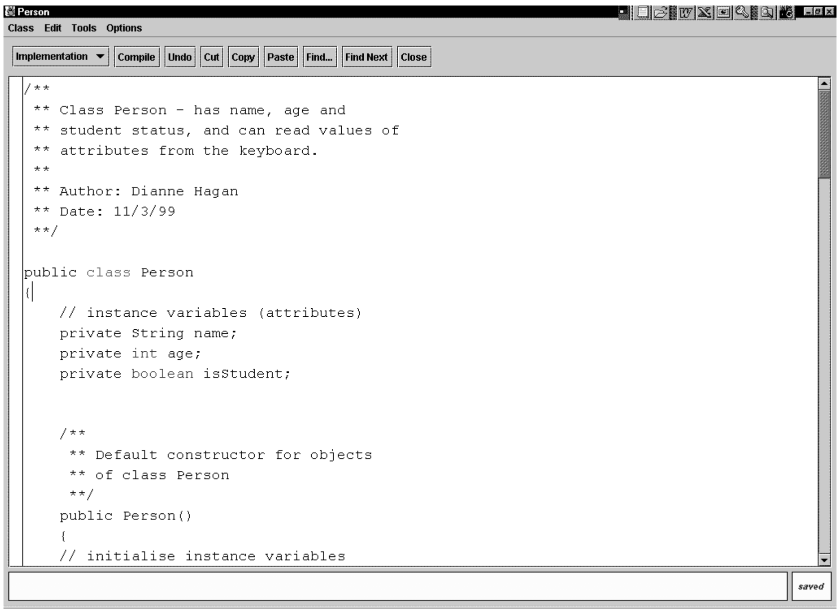


Figure 1: A class with some objects instantiated from it.

Difficulties Encountered by Students

Students were expected to download the software from the BlueJ website (Kölling, 2000) and install it at home. They had some practice during lab sessions in doing this, but unfortunately the environment in the labs was not exactly the same as in their homes, so some of them experienced difficulties. There were a few new versions of BlueJ released during semester, and students had to reinstall the software each time. This annoyed them, but was actually good practice. We found that many students had trouble installing the BlueJ software because the version we were using required them to change the path and the classpath. Most students had no experience of DOS, and even when they were given detailed instructions, some were unable to follow them. The latest version of BlueJ does not require students to do this. In many cases, when students had installed BlueJ correctly, an error message informed them that they were out of environment space. They then had to change the memory configuration of the machine. This was not difficult for them to do when it was explained to them.

For the first few weeks of semester, when they were writing only single classes, students were generally quite happy with BlueJ. After that, they were required to write a sizeable program by themselves, and some began to blame BlueJ for any problems they were having with programming. Students with experience in programming appreciated the benefits of BlueJ and defended it, but without convincing the others. In week 10, we decided to show them the alternative. We demonstrated how to use the JDK to write the programs that the students had already written, and had them practise using it to write more programs. After that, there were very few complaints about BlueJ; on the contrary, some students asked anxiously whether they would be allowed to use BlueJ in the following subject.



```
/**
 ** Class Person - has name, age and
 ** student status, and can read values of
 ** attributes from the keyboard.
 **
 ** Author: Dianne Hagan
 ** Date: 11/3/99
 **/

public class Person
{
    // instance variables (attributes)
    private String name;
    private int age;
    private boolean isStudent;


    /**
     ** Default constructor for objects
     ** of class Person
     **/
    public Person()
    {
    // initialise instance variables
```

Figure 2: The editor window

Student Support Mechanisms

We provided students with many avenues of help during the semester:

- The BlueJ website (Kölling, 2000) could be used to download software, to report problems and to see what problems other people had reported and the solutions that had been given. There was a link to this BlueJ website from the subject website.

ß The subject website also had, apart from the usual materials such as handbook, lecture notes and weekly exercises, an anonymous feedback section that could be used to ask for help with programming problems, and to discuss relevant issues.

ß Some examples of code relevant to the assignment were on the subject website, e.g. a class demonstrating how to generate random numbers.

ß Tutors in the subject were rostered onto a helpdesk that was available about 20 hours per week, including some time each day and some after-hours times for the evening students. This made it unnecessary for students to wait until their own tutors were available, as anyone on duty would know what they were supposed to be doing and how they were supposed to do it.

ß Tutors and lecturers were also available in person and via email for personal consultation, although this option was rarely exercised because the helpdesk made it largely unnecessary.

ß We did prescribe a textbook but this was a little confusing as it used the JDK and its own input classes. Some of it was therefore irrelevant. We did not follow the order of the topics exactly, as our subject was driven by the assignment that students were writing and what they needed to know at a particular stage. However, the textbook did provide additional examples and alternative explanations of concepts.

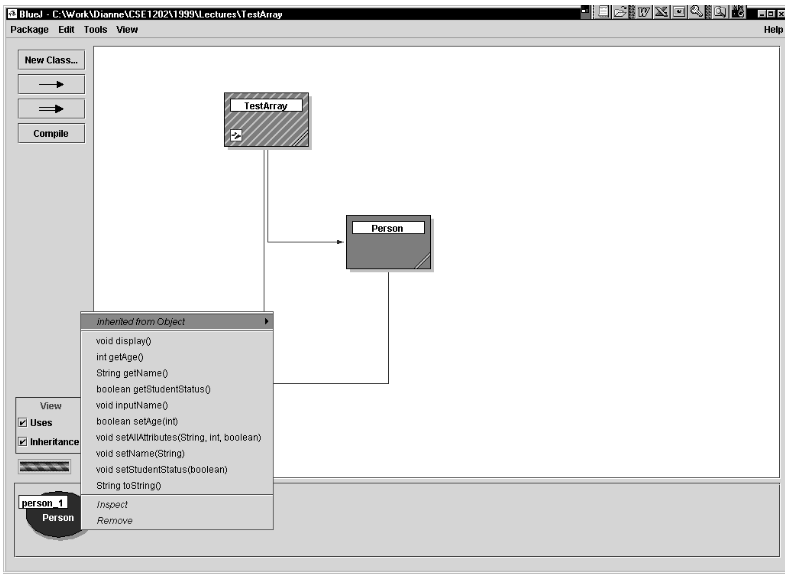
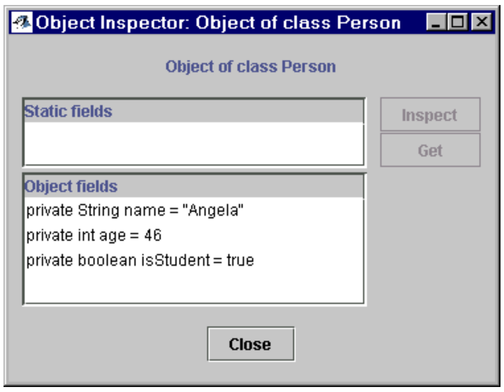


Figure 3: Asking an object to invoke one of its methods

Evaluation

A series of surveys during the semester were used to evaluate the students' perceptions about and attitudes to BlueJ. Students were asked to participate in these surveys, and we explained that this was not compulsory and that they were free to withdraw from the study at any time. Because we wanted a longitudinal study that would enable us to correlate students' results in a number of subjects with their reactions to BlueJ, we asked them to identify themselves when responding to the survey questions, by supplying their ID numbers. Students were assured that this data would be available only to a research fellow who was not involved in teaching, but it may be that they were still wary of identifying themselves. The majority of students in the class chose not to participate in the surveys, but we had enough respondents to be able to draw some conclusions about the usefulness of BlueJ as a learning environment.

There were four main stages in the research design for the evaluation:
1. A collection of basic background data on students, which included attitudes towards the subject and experience with programming languages.
2. Collection of intermediate data during the semester.
3. Email interviews for more in-depth data collection.
4. A final survey to explore the students' overall evaluation of the BlueJ experience.



```
private String name = "Angela"
private int age = 46
private boolean isStudent = true
```

Figure 4: The Inspector window showing the current state of an object

The survey forms were published on the subject website, and the data collected automatically. Tutors were asked to draw their students' attention to them in class at the appropriate times during semester, and ask them to respond immediately. Students were asked to comment on such issues as how much trouble they had had in installing BlueJ, how well they understood the difference between Java and BlueJ, the ease of using BlueJ, the friendliness of its user interface, how much BlueJ helped them to learn programming, and how their attitudes to BlueJ changed over the semester.

About a third of the students (a total of 120) participated in the survey, and the demographic data showed that they were roughly representative of the entire group in terms of proportions of male/female, local/international, full-time/part-time, etc. The biographic and demographic characteristics of the sample can be summarised as follows:

74% male
31% first language not English
32% international full-fee paying student
5% had no access to a computer outside those available at the university
44% direct from high school
32% from a diploma course at a non-university tertiary institution.

At the end of semester, the subject results of the participants were significantly better than those of the non-participants. We are not sure how to account for this. It may be that a random sample of students would have produced different survey results, but there is no way of knowing, as students could not be forced to participate. The survey responses show that the majority had a positive shift in attitude during the semester towards BlueJ as a learning environment and thought that it helped them to learn Java (see Table 1), although their opinion of its stability and reliability was comparatively low (see Table 2).

Rating	Week 6	Week 12
1 A great deal	12%	13%
2	12%	24%
3	30%	25%
4	22%	16%
5	8%	8%
6	8%	6%
7 Very little	8%	8%

Table 1: Student responses to "How much does BlueJ help you to learn Java programming?"

Responses to "What was the best thing about using BlueJ?" included:

- "I think the BlueJ is a cool programming environment."
- "Saves code automatically!!!"
- " .. I think you call it the workbench area. Additionally, being able to compile the objects and it does not allow to proceed, reporting any syntax errors. For a starter it a good tool for learning the concepts of programming / Java ."
- "There hasn't actually been a time when I said 'this is great'. But I think that it simplifies the whole concept of programming with Java. Makes us concentrate on the main parts of programming And the whole 'object oriented programming' issue is a bit more easily understood."

Responses to "What was the worst thing about using BlueJ?" included:

- "the error messages are not very helpful in the compiler"
- "the speed of BlueJ in general"
- "the download time for installing"

	Mean	Standard deviation
Overall rating	4.2	1.6
The interface	4.5	1.5
BlueJ and teaching Java	4.3	1.7
Stability of BlueJ	2.7	1.3
Downtime	2.9	1.4
BlueJ help with Java	4.3	1.8

Table 2: Student ratings of BlueJ item characteristics (scale of 1 to 7)

Conclusion

Complex research models are needed if comprehensive evaluation and development activities are to be undertaken effectively in educational innovation. If we had used an evaluation methodology that collected primarily qualitative data, it would have failed to give us information on the way BlueJ related to the broader educational attitudes and expectations of the students. If, on the other hand, we had used only quantitative methods, we would not have collected the important data on the specific factors that students found positive and negative about BlueJ. The early difficulties with installing and running the system may have distorted the results of the survey and the perceptions of students. For the first use of beta stage software, we considered the results very positive. By the end of the semester, as evidenced by the answers they gave to examination questions and in assignment interviews, students generally had a good grasp of object-oriented concepts:

ß they had none of the confusion about the difference between objects and classes which we had noticed in many students when teaching C++ in earlier years and which has been reported by others (eg. olland, Griffiths & Woodman, 1997);
ß they understood that each object has its own state;
ß they knew that it was necessary to instantiate a class and ask the object to invoke one of its methods in order to have that method executed.

Since this first subject was run, BlueJ has been improved and released as a non-Beta version. Installation of the system is now very much simpler. The BlueJ website still provides a mechanism to respond to problems that students encounter. We are currently using BlueJ in the introductory programming subject again, and are continuing our evaluation to see whether the improvements to the environment have had an effect on student perceptions, and to monitor the effectiveness of BlueJ as a learning tool.

References

Dann, W., Cooper, S. and Pausch, R. (2000) *Making the connection: programming with Animated Small World* Proceedings of 5th Annaul Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland, 41-44.

Gosling, J., Joy, B. and Steele, G. (1996) *The Java Language Specification* Addison-Wesley Publishing Company, Reading, Mass.

Holland, S., Griffiths, R. and Woodman, M. (1997) *Avoiding object misconceptions* Proceedings of 28th SIGCSE Technical Symposium on Computer Science Education, ACM, San Jose, California, 131-134, February 1997.

Jimenez-Peris, R., Patino-Martinez, M. and Parcios-Martinez, J. (1999) *Vis-Mod: a beginner-friendly programming environment* Proceedings of 1999 ACM Symposium on Applied Computing San Antonio, Texas, 115-120.

Kölling, M. and Rosenberg, J. (1996) *An object-oriented program development environment for the first programming course* Proceedings of 27th SIGCSE Technical Symposium on Computer Science Education, ACM, Philadelphia, Pennsylvania, 83-87, March 1996.

Kölling, M. *The BlueJ website.* [Online]. Available: http://www.bluej.org [13th September 2000].

Soloway, E.M. (1986) *Learning to program = learning to construct mechanisms and explanations* Communications of the ACM, 29, 850-858.