Orienting the Teaching of an Introductory Object-Oriented Programming to Meet the Learning Objective

Jasna Kuljis

The VIVID Research Centre, Department of Information Systems and Computing, Brunel University, West London, UK

This paper describes our experiences in teaching a first year object-oriented programming course. We used Java as a vehicle to teach programming principles and BlueJ as a Java development environment. The course was heavily supported by web-based resources delivered through WebCT. So far we consider the overall students' learning experience as being considerably enriched and a positive one.

Keywords: Web-based learning, teaching object-oriented programming, teaching Java.

1. Introduction

The majority of students, even those enrolled on computer science courses, find computer programming a difficult and complex cognitive task. Computer science educators have tried many methods to improve results in teaching programming. These include:

- the choice of the first taught programming language;
- the choice of the course textbook;
- the choice of programming environment;
- variations in teaching methods;
- variations in assessment.

A prescription has not been found on how to successfully teach programming or how to make any considerable improvement in a number of students successfully acquiring programming knowledge. Changing the language that is taught first does not significantly change the pass rate [1]. Even teaching programming using specially designed languages like, for example, Pascal, has not made any considerable difference. Neither does using different textbooks, nor does slowing the course down, nor does alternating a bottom-up and a top-down approach.

Some academics may have opted to lower standards and/or to reduce the quantity of material taught and concentrate on more 'important' topics [1].

Due to the increase in student population, classes of several hundred students for computer programming courses are not uncommon. For example, we normally have about 200 students in the programming class each year, however, last year we had just over 400 students in the class. Not surprisingly there is a huge variety in students' abilities, learning speeds, and attitudes. Since attention to difficulties faced by an individual cannot be easily addressed in a large class, we, educators, have to find an alternative way of helping students in their learning. Thus, we provide a variety of modes of learning and support different rates of learning.

Very little is known about the learning in general. It is not our intention to explore these complexities. More on that can be found in [1]. The paper describes a unique synergy between BlueJ [3] and WebCT [7] resulting in a rich environment created for our students.

The next section provides some background to our dilemmas and rationale behind the choice of

textbook, programming environment, and supporting web-based resources. The subsequent section describes the evolution of the introductory programming course at Brunel. A section describing a new approach to learning using WebCT and BlueJ follows. The next section gives a flavour of student adaptation to way of working. Finally the paper argues that, even though little is known about the learning of programming, the learning experience of students can be improved with extensive web-based support.

2. How to Handle Too Much Choice

The Department of Information Systems and Computing (DISC) at Brunel offers three undergraduate degrees leading to an award of BSc: Computer Science, Information Systems, and Applied Computing. All three degrees require students to take in their first year a programming course currently called "Construction of Programs". Traditionally students on that course do not perform as well as on other first year courses.

Before the start of each year, we review how the course went in the previous year and whether any changes are needed or desired for the following session. Since the success rate in the introductory programming course is never as good as in other courses, we continuously strive for improvements. Almost every year we make changes to our teaching, change course textbooks and choose on programming environment.

We have evaluated dozens of texts, and nearly every one is about how to program in Java, not how to construct programs using Java as the programming language. Also, most textbooks follow a standard procedural approach. Programming concepts typically consists of primitive types, control statements, loop statements, before going into more elaboration on classes, objects and methods. A typical first program is usually "Hello World" program, hardly an introduction to object-orientation.

Since using I–O in Java requires considerable knowledge, many textbooks come with special environments or with special classes with custom-written I–O methods. Students taught following this type of approach end up confused and very few grasp the fundamentals of object-orientation. Our experience shows that two of the most difficult concepts for students to understand are the difference between a class method and an instance method, and between a class variable and an instance variable (agreeing with [4]).

3. Legacy Courses

This section describes how introductory programming course at DISC evolved over the last three years.

In 2001-02 there were 2 hour lectures plus 2 hours of structured laboratory sessions (we call them 'labs') a week. Labs were used to put into practice what was taught during the lecture. This was a change from the previous year when there were only 3 hour lab sessions for each student and when all new material was taught within that session. The course was delivered by a teaching team consisting of 9 academics and 2 teaching assistants (PhD students). Teaching to each group of students was done by a different person who made their choice of how to present material and what to emphasise; therefore, each group of students had a unique experience.

In order to provide a common experience, we decided to separate lectures where new material is presented from the practical sessions where these were put into practice. Two tutors were responsible for each session. The main course textbook was Hortsmann [5]. All course material was made available on the dedicated course web page on the Department's Intranet. The course material included details about the course (e.g., study guide, timetable, teaching team contact details, software downloads, etc), copies of lecture slides, lab exercises set for each week, notices, coursework details, past exam papers, etc.

JBuilder by Borland was used as the programming environment and a custom-written library containing I–O methods was written by a colleague Tony Elliman. All coursework that was finished on time was collated electronically (file transfer). However, late coursework was handed in on floppy disks. We used software specially designed by Tony Elliman for marking programming projects written in Java. For each coursework the marking software was provided with a script that contained details of the marking scheme and the software guided the marker

through a standard set of questions and answers and tests. The exams were in a form of multiple choice questions and were marked using scanning software.

Again, the results were disappointing and there were many concerns that students were not able to program. Our main hypothesis was that since students were expected to learn how to program in the first semester, in order to spend more time on algorithms in the second semester, their time to learn programming was too short.

In 2002-03 the overall syllabus remained very similar, apart from more emphasis being given to general programming principles and problem solving rather than to data structures and algorithms. There were 2 hour lectures plus 2 hours of structured lab activities a week. The teaching team consisted of seven academics and five teaching assistants. Lectures were given in one session to all students enrolled on the course. Students were divided into nine groups and each had a 2 hour lab session per week. Two tutors were responsible for each session. The main course textbook was Hortsmann [6]. Again, all course material was made available on the dedicated course web page on the Department's Intranet. We again used JBuilder and in-house software library.

The assessment consisted of 2 programming assignments and the exam. All coursework that was finished on time was submitted electronically (file transfer) and, again, late coursework was handed in on floppy disks. We again used marking software. The exam was the form of multiple choice questions and questions were divided into three sections. The first section tested the knowledge required to pass. The second section tested the knowledge required to get a better grade (up to grade B) and the third section was used to differentiate between better students (for grade A). Again, the exams were marked using scanning software.

There were more than 400 students on the course and coordination problems were immense especially those related to the assessments.

4. Web-Enhanced BlueJ Learning (WEBL)

Our new approach WEBL uses WebCT to facilitate constant students' progress learning object-oriented principles with BlueJ. This required drastic changes in 2003-04 which include:

- the new textbook
- the new programming environment
- different web-based support
- different assessment

The following sections list the reasons for changes made.

4.1. Objects First and BlueJ

The reason for changing the textbook was the fact that the majority of students were using Java as a procedural language because they could not understand object orientation. They were not solving problems by designing classes, but were writing monolithic programs using only the main class. They could not separate the two main concepts, class and an instance of the class. Other important object-oriented concepts like inheritance and polymorphism was a mystery for them. We wanted students to truly understand the concept of objects, but it is difficult to put this into practice in Java. Many hurdles such as of syntax and detail have to be overcome before learners can have first experience with living objects [2].

We used JBuilder by Borland as the main programming environment for many years, but it was not learner friendly.

The combination of a textbook by Barnes and Kölling [2] and the BlueJ development environment [3] overcame the main weaknesses of previous textbooks and the programming environment. Barnes and Kölling [2] focus on the general object-oriented and programming concepts, integrating this with the use of the BlueJ environment. BlueJ is an interactive development environment designed explicitly as an environment to teach introductory object-oriented programming by Monash University, Australia. The main strengths of BlueJ are [2]:

- It provides visualization of class structure. It automatically displays a UML-like diagram representing the classes and relationships in a project (see Fig. 1).
- Users can directly create objects of any class (see Fig. 2) and then interact with methods (see Fig. 3). They can try out a method immediately after it has been written, without the need to write test drivers. This facility is an invaluable aid in understanding the underlying concepts and language details.

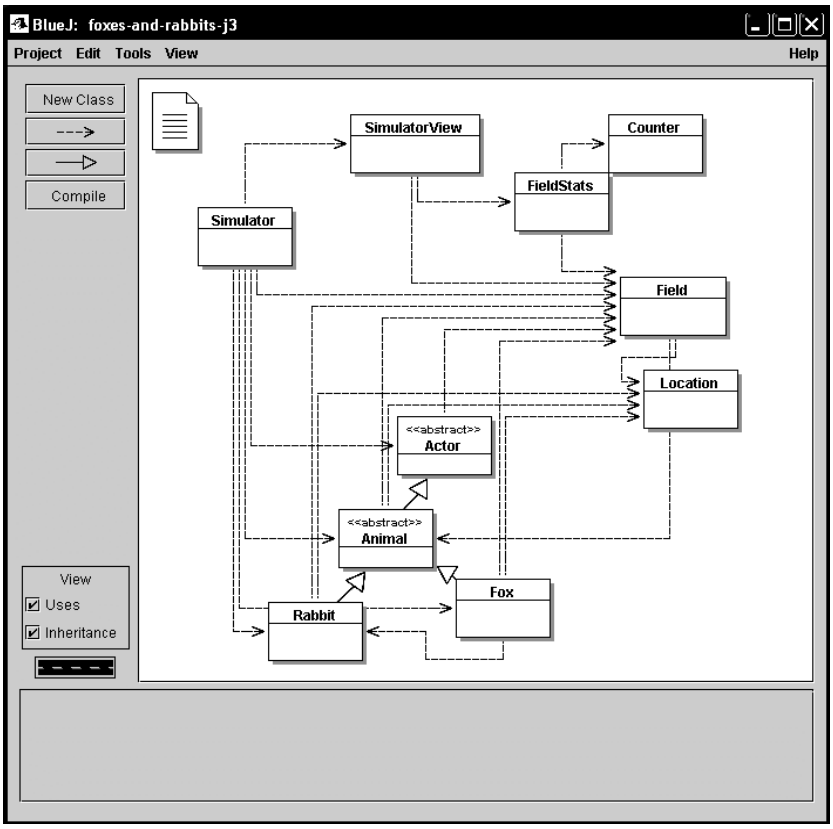

Fig. 1. Class diagram in BlueJ environment.

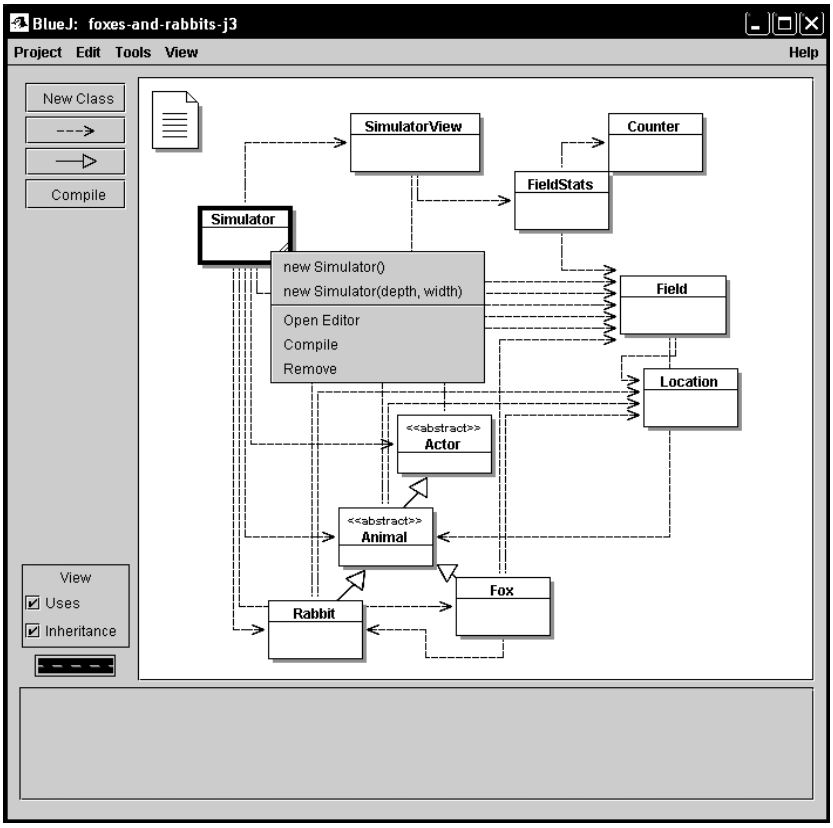

Fig. 2. Creating an object in BlueJ.

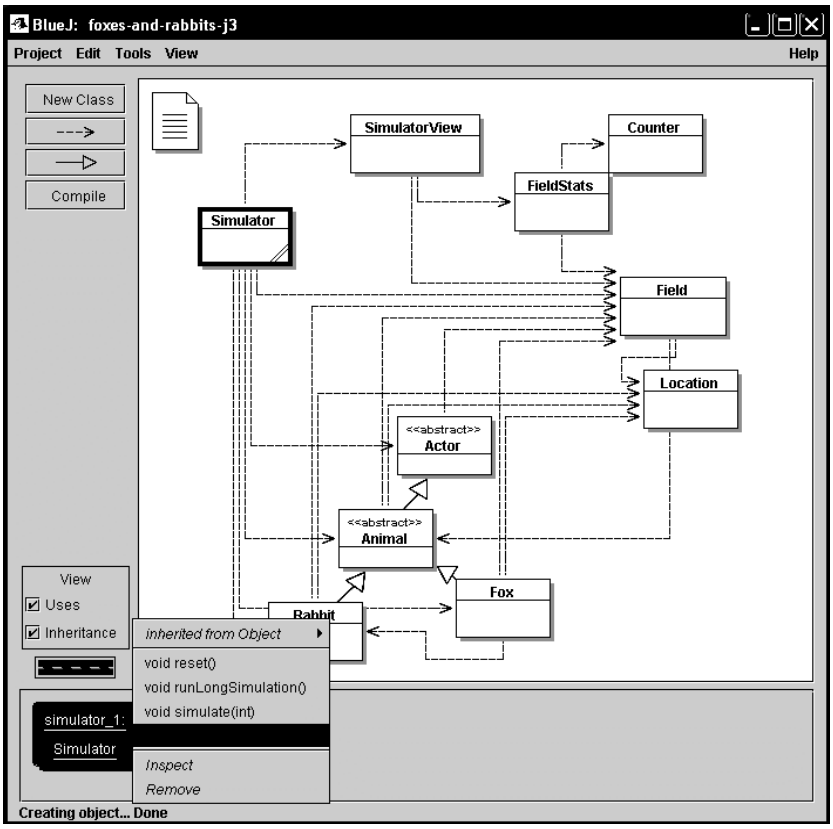

Fig. 3. Calling a method on an object in BlueJ.

4.2. WebCT

As already mentioned, the course had a strong web resource base. However, this was a static web page that had several weaknesses. The main weaknesses were the following.

The course web page was Intranet-based and not easy to access, especially through external means. Students were able to read material on the web, to copy it, to print it, and to download software. There were no facilities for interaction like, for example, to post queries, to submit their work, and to create a discussion forum. So students used e-mails as the main communication channel. With 400 students on the course, a considerable and time consuming effort was spent answering similar questions.

We therefore migrated to WebCT, because it provides the following facilities:

- access from any Internet point;
- authentication using userid and password;
- it allows posting files;
- submission of student work;
- quizzes and tests;
- course communication using course notices and discussion forum.

Different users of WebCT have different views and different access rights. The main categories are the designer, teaching assistants, and students. Designers have access to all parts of a particular course web page (see Fig. 4) and they can

- create web pages for the course (see Fig. 5);
- upload various types of files and organize them into directories;
- create tests and quizzes;
- create submissions;
- create a class list;
- give access rights to students and teaching assistants;
- view and download work submitted by students;
- mark submitted students' work and provide comments on WebCT;

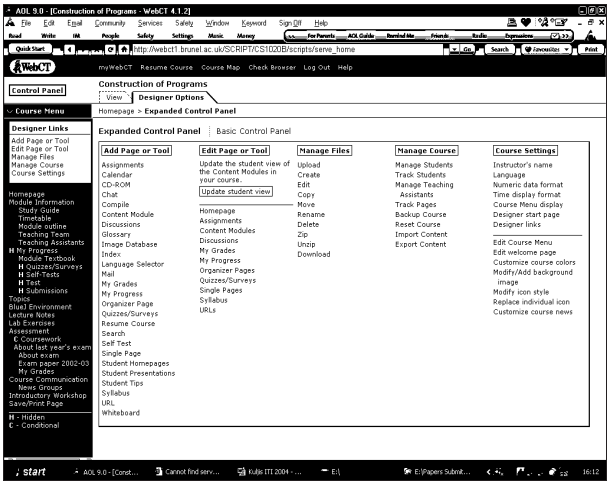

Fig. 4. WebCT – designer's view.

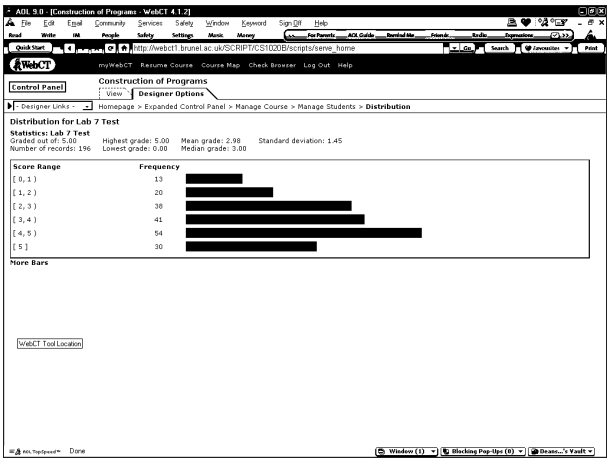

Fig. 6. Graph representing a lab test's scores.

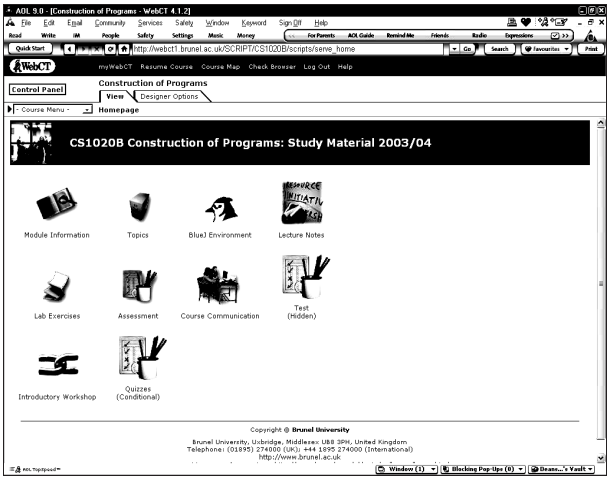

Fig. 5. WebCT course home page.

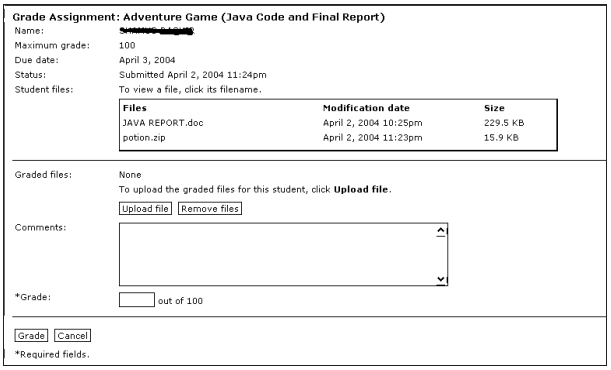

Fig. 7. A coursework submission made by a student.

- modify marks for student tests and submitted coursework;
- view statistics on overall submissions and tests (see Fig. 6);
- view statistics on students' access to WebCT resources like dates, pages accessed, submissions made (see Fig. 7), attempted tests, etc.;
- post course notices;
- reply to students posted queries in the discussion forum.

Teaching assistants can view all WebCT resources and have access right to all facilities related to students and their assessment, including adding students to the course list. They can also view all statistics and participate in the discussion forum.

Students can view all public material on WebCT, plus they can do the following:

- attempt tests and quizzes;
- submit their work by uploading files on WebCT;
- view only their own marks for completed tests and the feedback on their work;
- participate in the discussion forum.

4.3. Assessment

In the previous years our standard assessment consisted of 2 coursework assignments (50%) and the exam (50%). However, we have become increasingly aware of the number of very similar (sometimes identical) programs submitted by different students. It is very difficult to trace who the original author was and who were

the students copying the work. Low turn-out for lab sessions and very low Java literacy at the end of the course were things we needed to improve on.

We therefore decided to somehow assess the work done as a part of lab session exercises. Each week students are given exercises (made available on WebCT) that address the material taught in that week to undertake during that week's lab session. There are 22 lab sessions in the course. When appropriate, we made tests related to these labs. Students were asked to undertake the corresponding lab test upon the completion of lab exercises (see Figure 8). Each test was only made available for two weeks. Sixteen of these tests identified at the beginning of the course are used for the assessment. Lab tests contribute 25% of the overall mark.

We kept one coursework programming assignment contributing another 25% to overall mark. But, instead of giving a highly specified problem for students to program as in previous years, this year we gave our students the opportunity to develop their own specification of a very general problem (a text game) and then to implement it. The examination is a combination of multiple choice questions (maximum grade for this part is B) and problem solving questions to which students will be expected to provide their own answers (this part can get them to A grade provided that they already scored B on the multiple choice part).

5. WEBL Working

The teaching team consists of five academics, one of whom is the course leader, and ten are teaching assistants. The course leader is responsible for the course organisation and management. A course organisation involves:

- Planning the course well ahead for each academic session;
- Clear division of duties and allocation of responsibilities;

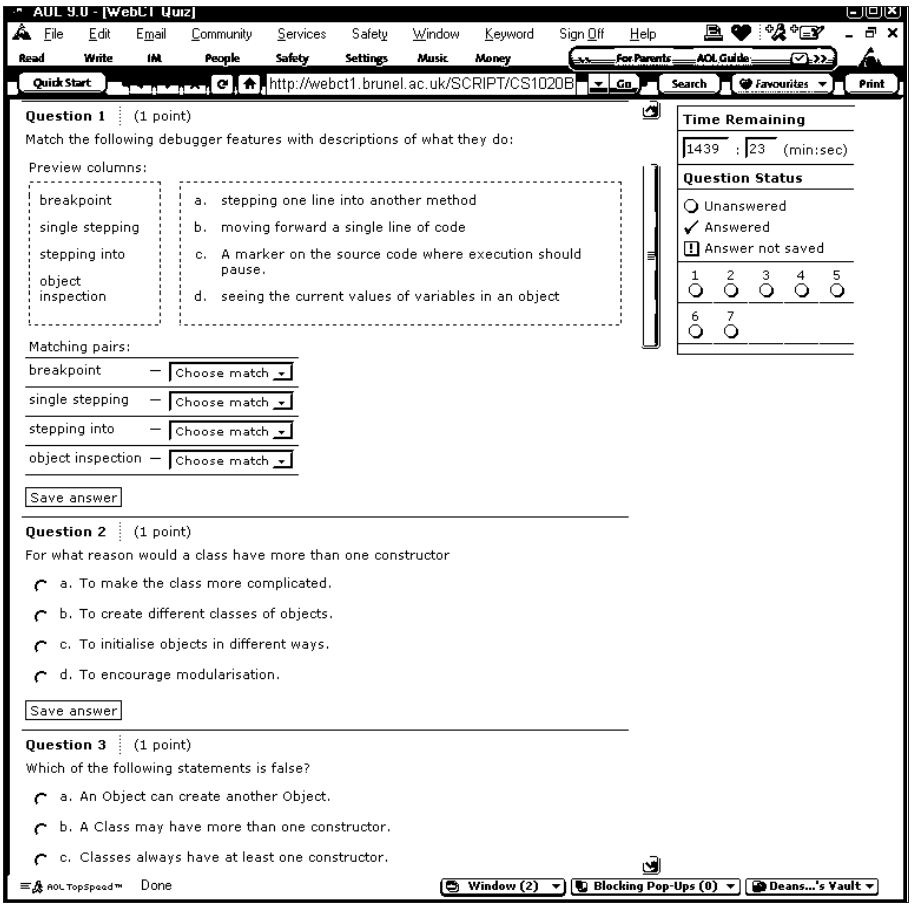

Fig. 8. A lab test on WebCT.

- Regular weekly meeting of the whole teaching team so that any problems can be acted upon and to provide a consistent approach by all team members;
- Coordination of activities.

There was considerable effort put into migrating to WebCT, but that effort is already proving worth while. The number of emails sent to tutors has dropped with students using the discussion forum to post their questions, cries for help, etc. Now there is a lively community of students helping each other and engaging in constructive dialogue about the programming problems.

With the on-line assessment of lab tests students are attempting, almost all of them, which in its turn requires students to be up to date with the material taught. However, unexpectedly, turn up for the lab sessions has not been high. On investigation, the main reasons for that are as follows:

- programming concepts are taught in the lectures and we are using the textbooks;
- majority of students have their own PCs;
- lab exercises are published on WebCT and can be accessed from any Internet access point;
- BlueJ is free, so is Java compiler;
- lab tests are available for two weeks and are on WebCT so can be accessed through Internet.

So students are doing work, as indicated by the high number of students attempting each lab test. Thus we have succeeded in encouraging them to keep up to date with the course.

We observed during lab sessions that students have less problems understanding the difference between classes and objects. Also, compared to previous years, they are much more inclined to solve problems by designing classes. So far, there have been no complaints concerning lack of support or the transparency of the process.

6. Conclusion

This paper describes WEBL a unique approach combining the strength of BlueJ and WebCT to facilitate constant students' progress learning object-oriented principles with BlueJ.

Early indications are encouraging both in terms of the quality of students' discussion and in the test results. The main achievement up to date is an unusually high level of students' involvement and their sense of belonging to a community willing to help each other. For the teachers, teaching programming has become pleasurable with students keeping up to date and, as far as we can tell, learning subject.

References

[1] BALDWIN, L. AND KULJIS, J., *Visualisation Techniques for Learning and Teaching Programming*, Journal of Computing and Information Technology, 2000. **8**(4): pp. 285–291.

[2] BARNES, D. J. AND KOLLING, M., *Objects First with Java. A Practical Introduction Using BlueJ*, Harlow: Pearson, 2003.

[3] BlueJ, `http://www.bluej.org`.

[4] HONG, J., *The Use of Java as an Introductory Programming Language*, ACM Crossroads Student Magazine, The ACM's Electronic Publication, 1998(4.4), `www.acm.org/crossroads/xrds4-4/introjava.html`.

[5] HORTSMANN, C., *Computing Concepts with Java Essentials*, New York: Wiley, 1998.

[6] HORTSMANN, C., *Big Java*, New York: Wiley, 2002.

[7] WebCT, `http://www.webct.com`.

Received: June, 2004
Accepted: June, 2004

Contact address:
Jasna Kuljis
The VIVID Research Centre
Department of Information Systems and Computing
Brunel University
West London
Middlesex, UB8 3PH, UK
Phone: +44 1895 203 081
e-mail: `Jasna.Kuljis@brunel.ac.uk`

JASNA KULJIS is a Reader in the Department of Information Systems and Computing at Brunel University. She gained her Dipl.Ing. degree in theoretical mathematics from the University of Zagreb, Croatia. She gained her M.S. in information science from the University of Pittsburgh, USA, and the Ph.D. in information systems from the London School of Economics, University of London. Her current research is in human computer interfaces. She is mostly interested in the design of graphical user inter aces and in the development of new paradigms that would further enhance the usability of interactive computer systems. She is Director of the VIVID Research Centre at Brunel University. Dr. Kuljis' email and web addresses are `jasna.kuljis@brunel.ac.uk` and `www.brunel.ac.uk/~csstjjk`, respectively.