

AC 2008-1310: TEACHING JAVA – OBJECTS FIRST WITH BLUEJ

Xuemin Chen, Texas Southern University

David Olowokere, University of Alabama at Birmingham

Graham Thomas, Texas Southern University

Teaching Java – Objects First with BlueJ

Abstract

The traditional way to teach computer languages such as C, C++ is to start with a simple program “Hello, world!”. Java derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. But Java uses the object-oriented programming methodology. To start teaching “Hello, world!” will not help students understand the object-oriented programming concept of Java. Therefore, it will be difficult for students to take advantage of object-oriented programming concepts. In this paper, an objects first Java teaching method with BlueJ, a simplified and virtualized development environment, is presented. A post-course assessment is conducted. The interpretation of the assessment results is also discussed.

Introduction

Java was created by James Gosling at Sun Microsystems¹. It combines object-oriented (OO) features such as data abstraction, inheritance, and dynamic binding with procedural features such as variables, assignment, and control structures. The result is a powerful but complex language that is difficult for beginning programmer to master.

The Java programming language has become increasingly popular in recent years because of its support for the platform independent, and the OO paradigm. Teaching Java is becoming more and more common in university departments². There are two ways to teach OO computer languages such as Java³. The traditional way (objects later) is to start with data type, then control structures, eventually classes and objects. An alternative way is the ‘objects first’ approach, which is to start first with classes and objects, then move to source code. Java is not the introductory computer programming language for students at authors’ department. The students already took C++ in freshman or junior years. The students were taught C++ programming language concepts and skills without the object-oriented paradigm.

Java derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java programming language platform is based primarily on object-oriented structures. This makes the use of the language as a teaching tool much easier when teaching object-oriented concepts. The way to understand the OO paradigm is to deal with the fundamental concepts - classes and objects. The longer this is left, the more difficult the paradigm shift becomes.

The transition from the procedural programming paradigm to OO paradigm is very difficult⁴. Object-oriented development requires a new way of thinking⁵. The syntax-driven approach (objects later) can take students’ attention away from the underlying concepts and principles of OO^{6,7}. In the computing education community, a well-known educational design pattern exists that states that important concepts should be taught early and often⁸. Based on these concerns, the questions bringing which needs consideration are as follows:

- Which textbook is suitable?
- Which programming environment will be used?

- What kind of teaching strategy should be adopted?

The search for a suitable textbook started with these keywords: Java and object first. The textbook, *Objects First with Java – A Practical Introduction Using BlueJ*⁹, was selected. Actually, this textbook is broadly adopted in education community^{10, 11}.

Software tools for programming in Java can be equally overwhelming to beginning programmers. Professionally integrated development environments (IDEs) like Eclipse¹², and Borland JBuilder¹³ provide programmers with many features for editing, compiling, and debugging programs, but their increased sophistication makes them less user friendly and difficult to use. Beginning programmers are consequently faced with two challenging tasks, learning the IDE and learning the language. A simple IDE that can allow students to visualize objects and their relationships would be ideal. BlueJ¹⁴, written in the Java programming language, is that kind of IDE developed to support learning and teaching.

The next section reviews the BlueJ integrated development environment. The subsequent section describes our Java teaching objective and procedure. A section describing the final project follows. The next section gives the post-course assessment and interpretation of the assessment results. Finally a conclusion is presented.

The BlueJ IDE

The development of BlueJ was started in 1999 by Michael Kölling and John Rosenberg at Monash University in Melbourne, Australia, as a successor to the Blue system¹⁵. Blue was an integrated system with its own programming language and environment. BlueJ implements the Blue environment design for the Java Programming Language. It has become a popular choice for introductory programming courses because of the ease with which absolute beginners can learn how to use its features^{9, 10, 11}.

BlueJ presents on screen a graphical overview of a project structure in the form of a UML (Unified Modeling Language) like class diagram as shown in Figure 1. It then allows the interactive creation of objects from any given class in a software project. Once an object has been created, it becomes visible to the user and any of its public methods can be interactively invoked by selecting it from a pop-up menu. Parameters and method results are entered and presented through dialogue windows. In particular, using the *Inspect* option of the pop-up menu associated with objects, students can directly see the values of the fields of an object. This allows them to immediately see the effect of a method invocation on that object and also simplifies the debugging process.

The environment is carefully designed to be very simple to use. The goal is that students do not need to spend significant time struggling with the environment, but instead concentrate on the programming task. This was achieved by a conscious trade-off: Much of the functionality presented in other environments such as Eclipse and JBuilder is not included in BlueJ. This makes BlueJ mainly for educational purposes, but also suitable for small-scale software development.

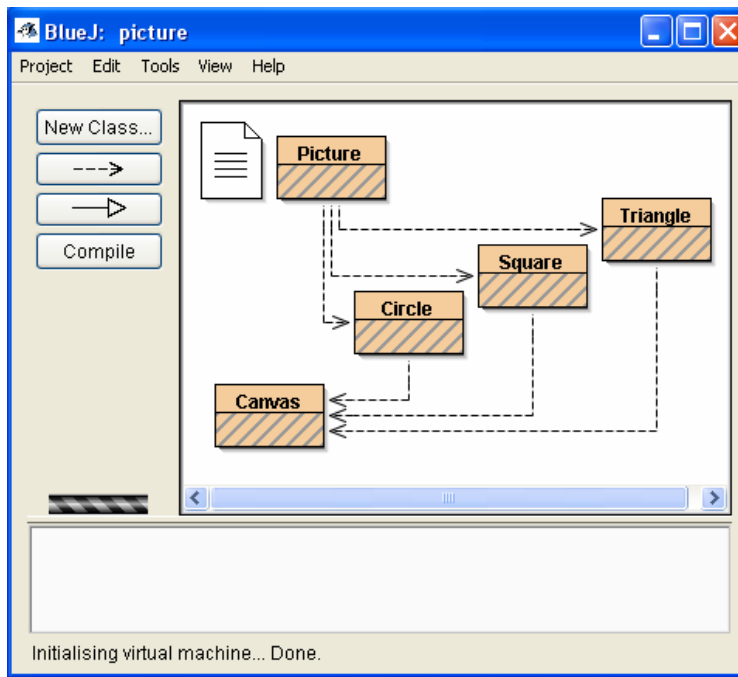


Figure 1 The BlueJ main window

The Course Objective and Teaching Procedure

Cited from curriculum, the course title is “Java Programming”, and the short description is “High-level, object-oriented language programming using JAVA. The course includes inheritance and polymorphism, implementing hiding, and the creation of JAVA applets for internet usage.” Upon completing this course, students should be able to program in Java at an intermediate level. Specifically, they should be knowledgeable about object-oriented programming, and able to implement a median software system in Java.

The Computer Engineering Technology is a new program in author’s Engineering Technologies Department. This programs has begun the process for seeking ABET accreditation in the near future. The proposed roles of Java course in aiding program outcomes are a, b, d, e, I and k defined in ABET Criterion 3.

The BlueJ IDE is introduced in the first class, examples on project “Shapes” are demonstrated; and examples are used to explain the very important concepts – Object and Class. Thereafter, a hands-on laboratory task follows. With this simplified and virtualized IDE, it is expected that most students will understand the concepts of object and class in Java programming structure within the first few weeks.

In order to comply with the laid-down objective of the course, and that is ensuring that students can program in Java at an intermediate level after course completion, the examination method for the course has been greatly modified. The first exam is closed book. In this exam, we focus on Java concepts and syntax. The second exam is open book. The students will demonstrate their

programming capability; the intent being to allow students to demonstrate their programming skills using their textbooks or other materials as references, rather than taking an objective test about Java programming concepts.

As mentioned in the introductory section, the Java Programming course is not the introductory computer programming language for students at the authors' department. The students would have taken C++ programming prior to the Java class. On completion of topics related to foundations of object orientation, such as objects, classes, abstraction, modularization, flow control, library, responsibility-driven design, coupling, cohesion, refactoring; and some complex concepts such as inheritance, polymorphism, exception handling and implementing hiding; students are then taught the graphic user interface (GUI) and Applet programming skills.

There have been some arguments on the teaching of OO concept through GUI programming. An opinion is that students spend excessive time moving and placing GUI components around the screen, and those OO concepts can be easily embedded or completely ignored. Another opinion is to teach OO programming effectively and give students firsthand experience with its benefits, one needs a substantial framework. And the GUI programming provides a particularly effective vehicle for this purpose because it is relevant to virtually all applications and provides immediate feedback on the correctness of OO structures through tangible, visual results¹⁶. In our case, the students show a lot of interests in learning these concepts. From the students' point of view, learning the concepts of GUI and Applet gives them insights into industrial application.

Instead of a final exam, students are assigned a final project which enables them to apply their programming knowledge, to learn how to function effectively on teams and to develop self-learning capability.

When designing the final project, the main concern was to design a project which would incorporate most of the concepts learned during the semester; and with the project presenting numerous opportunities for useful application. In the same semester, one of the authors was teaching the DC Circuits laboratory. The freshman students have some difficulty in remembering the resistor color code in the DC Circuits class (taught by one of the authors of this paper) and in using the knowledge to read resistor. There are a lot of graphic resistor color code calculators on Internet. For example the one developed by TI¹⁷. When the user changes the resistor color band, the resistor value will be recalculated and displayed on screen. But that kind of calculators don't have quiz functionality to test the students' color code reading capability. Hence, it was decided to use this as the final project.

Another concern is the virtual and remotely accessible Laboratory development in authors' department. Computer based learning has become an important part of education. The Internet (Website) has become a widespread tool for teaching and learning. The Website enables more flexible delivery (anytime), distance education (anyplace), new visualization possibilities (interactivity), and cost reduction. One of the very successful virtual laboratories is the iLab developed by MIT^{18, 19}. The web-based laboratories are impossible to totally replace the traditional laboratories. But its relatively low cost, flexibility, and remote accessibility will dramatically improve the teaching and learning capability. The resistor color code lab is the first

one we developed for this purpose. More labs related to DC/AC circuit, control, DSP, data communication and so on will be developed. This plan will provide us lot of project ideas.

The Final Project

The final project is to build a resistor calculator with Java GUI and Applet. The project tasks are listed as follow:

1. Use Java applet to develop a GUI that is similar to the reference webpage¹⁷.
2. Display the resistor image.
3. Make the color band change when selecting the different colors in the Como Boxes.
4. Make the webpage have two modes: learning mode and quiz mode. If the user selects the learning mode, the webpage function is similar to the reference webpage. When the user picks the quiz mode, the computer randomly generates a color band, and the user can input the resistor value into the textbox. A SUBMIT button is built for the user to submit the value. The computer displays a message: “Correct if the value entered by the user is right” and “Wrong if the value entered by the user is incorrect”.

This project covers most of the topics in the textbook⁹. The project is done as shown in Figures 2 and 3.

As we discussed in a previous section the BlueJ IDE is a simple and effective teaching tool and also good for small scale software design. Implementation of the project was done using the professional IDE, Eclipse as this was more suitable for purposes of this project.

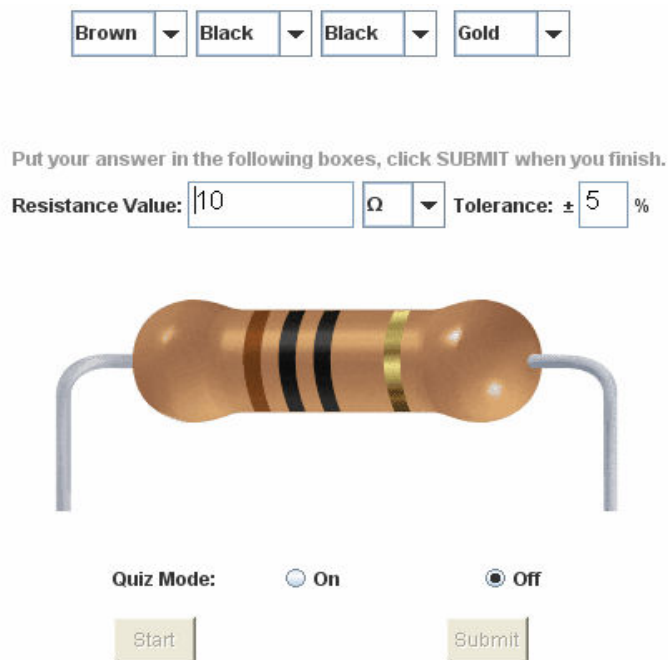



Figure 2 The default mode is quiz off. Resistor value is recalculated as the band changing

Orange Green Gold Red

Put your answer in the following boxes, click SUBMIT when you finish.

Resistance Value: Ω Tolerance: \pm %



Quiz Mode: On Off

Figure 3 The quiz mode is on. The user read the color code, fill the box. Then submit for judging.

Post-Course Assessment

This is a small class since the program is new. There are only ten students, and they transfer from other majors to the Computer Engineering Technology program. One student did not complete class due to some personal reason. The remaining nine students completed the course and did the course survey. The following are the survey questionnaires. Number of students selecting each answer are indicated.

1. In C++ Programming, we start from the “Hello, world!” example. In Java Programming, we start from classes and objects. Which way do you prefer for Java learning?
 - a. Objects first 6
 - b. Objects latter 2
 - c. Don’t know 1
2. Java vs. C++: learn
 - a. Java is Harder 4
 - b. Unsure 1
 - c. Java is Easier 4
3. Java vs. C++: program
 - a. Java is Harder 4
 - b. Unsure 1
 - c. Java is Easier 4

4. Do you confuse classes with objects when you finished this course?
 - a. Yes 1
 - b. No 6
 - c. Don't know 2
5. Do you like the IDE (Integrated Development Environment) provided by BlueJ?
 - a. Yes 7
 - b. No 2
 - c. Don't know
6. Do you own the textbook?
 - a. Yes 7
 - b. No 2
7. Is the textbook helpful?
 - a. Yes 6
 - b. No 1
 - c. Don't know 2
8. Would you recommend this course to your fellow students?
 - a. Yes 9
 - b. No 0
 - c. Don't know 0

Though the sample size used is small and the survey results might not reflect the whole picture valuable information can be obtained from analyzing the survey results. 67% of students preferred the objects first teaching method. It was interesting to note that half of the students think Java learning and programming are harder than C++. Java derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. It does not mean that the Java is much easy to learn and program. Most of the students (67%) don't confuse the class with the object when they finish this course. Two students were confused. This was due to their poor attendance. 78% of students enjoy using the BlueJ IDE as the learning tool. The trade-off of the simplified IDE is that it doesn't provide powerful help as the professional IDEs do. The textbook⁹ provides many examples that are easy to follow. 78% of students have the textbook and 66% of students think the textbook is helpful. The good thing is all of students gain their programming capability and recommend this course to their fellow students. In the comments area, lots of the students mention the final project. They like this project and learn a lot from it. They also mentioned that the project was a bit difficult. To enable them to complete the project the instructor provided additional assistance.

Conclusion

Java as an object-oriented computer language is not easy to teach. Although not taught as an introductory computer language in authors' department, student students still find that Java is a difficult object-oriented computer language. However, with right teaching tools and methods the teaching outcomes can be significant. BlueJ which is designed with the unique features of interactivity, visualization and simplicity can create a good first impression on students. A professional IDE such as Eclipse is recommended for complicate project design. A well designed final project with graphic user interface will motivate the students to learn and help them to better understand Java.

Bibliography

1. James Gosling, Henry McGilton, The Java language Environment: A white paper, Sun Microsystems, 1996
2. M. Kölling, J. Rosenberg, Guidelines for Teaching Object Orientation with Java, Proceedings of 6th conference on Information Technology in Computer Science Education (ITiCE 2001), Canterbury, 2001.
3. Albrecht Ehlert, Carsten Schulte, Learners Views on Objects-First and Objects-Later – Results of an Exploratory study, 11th Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts, Berlin, Germany, July 2007.
4. Jürgen Börstler, Marie Nordström, Lena Kallin Westin, Jan-Erik Moström, and Johan Eliasson, Transitioning to OOP/Java - A Never Ending Story, http://www.spop.dk/final-chapters/Transitioning_to_OOP.pdf.
5. V. Bacvanski and J. Börstler. Doing Your First OO Project: OO Education Issue in Industry and Academia. Proceedings of OOPSLA Addendum, Atlanta, GA, USA: ACM Press, 1997.
6. J. Bennedsen and M.E. Caspersen, Model-Driven Programming, in The SPoP book, J. Bennedsen, M.E. Caspersen, and M. Kölling, Editors. 2006.
7. M.E. Caspersen and H.B. Cristensen, CS1: Getting Started, in The SPoP book, J. Bennedsen, M.E. Caspersen, and M. Kölling, Editors. 2006.
8. J. Bergin, Fourteen Pedagogical Patterns for Teaching Computer Science, Proceedings of the Fifth European Conference on Pattern Languages of Programs (EuroPLop 2000), Irsee, Germany, July 2000.
9. David J. Barnes and Michael Kölling, Objects First with Java – A Practical Introduction Using BlueJ, Third Edition, Pearson Education, 2006.
10. Jasna Kuljis, Orienting the teaching of an Introductory Object-Oriented Programming to Meet the Learning Objective, Journal of Computing and Information Technology –CIT 12 (2), 135-142, 2004.
11. Stelios Xinogalos, Maya Sartatzemi, Vassilios Dagdilelis, and Georgios Evangelidis, Teaching OOP with BlueJ: A Case Study, Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06).
12. Eclipse, <http://www.eclipse.org>.
13. JBuilder, <http://www.borland.com/jbuilder>.
14. BlueJ, <http://www.bluej.org>.
15. BlueJ, <http://en.wikipedia.org/wiki/BlueJ>.
16. Jesse M. Heines and Martin J. Schedlbauer, Teaching Object-Oriented Concepts Through GUI Programming, Eleventh Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts, Berlin, Germany, 2007.
17. TI Four Band Resistor Color Code Calculator, <http://focus.ti.com/docs/toolsw/folders/print/4rescolorcalc.html>.
18. Gerardo Viedma, Isaac J. Dancy, and Kent H. Lundberg, A Web-Based Linear-Systems iLab, American Control Conference, June 8-10, 2005. Portland, OR, USA, pp. 5139-5144.
19. iLab, <http://icampus.mit.edu/iLabs/>