Submission Title:

OOPS as a tool for teaching object-oriented graphics in Java

Contact information:
Robert McCartney robert@cse.uconn.edu
Kate Sanders KSanders@ric.edu
Ruth Ungar ruth@engr.uconn.edu
Andries van Dam avd@cs.brown.edu

Problem statement

We are teaching courses in object-oriented design and programming, concentrating on graphics applications in Java. The standard Java API classes plus Swing and Java2D are very complex for beginning students to use and (especially Java2D and Swing) violate many of the object-oriented principles that we want to teach.

Solution Overview

We are using a package, OOPS (written by John Goodwin at Brown) that provides fourteen classes and six interfaces. Several of the classes are abstract; the classes the students use directly are `Frame, Ellipse, Rectangle, RoundedRectangle, Line, Image, QuitButton,` and `ConversationBubble`. Both students and instructors learn these eight classes quickly and easily.

OOPS does all of the necessary housekeeping so that whenever one of these objects is instantiated, it appears in a drawing area on screen. As multiple shapes are instantiated and displayed, the OOPS `Frame` class stores these in a collection, automatically does screen updates, and allows graphics shapes to be treated like objects: color, for example, is an attribute of a shape that can be accessed and changed using get/set methods with no need to change the color attribute of the Graphics instance during repaint.

Additionally, the shape objects all define individual mouse methods (`mousePressed, mouseReleased, mouseDragged,` and `mouseClicked`), so one can add the ability to interact with a graphic shape simply by extending one of these classes and overriding the appropriate mouse methods, without

the overhead of defining and registering listeners.

Experience with the Solution

OOPS has been used at three places: at Rhode Island College and Brown University since Fall, 2002, and at the University of Connecticut since Fall, 2003. The total number of students taught has been about 100 at RIC, 120 at UConn, and 125 at Brown.

Brown is in the process of making a transition to OOPS from a more sophisticated library, NGP (see Java Task Force submission, "NGP as a tool for creating Object-Oriented, 2D graphical applications in Java"). The two libraries are fundamentally similar, but OOPS is smaller than NGP. NGP is suitable for courses that want to use a library throughout; OOPS is designed for use in the early part of the semester, in courses where a transition to "real" Java is planned.

Using OOPS, students are able to write simple graphics programs (to display a red circle on the screen, e.g.) during lab the first week. Soon they create cartoons made from graphical shapes and conversation bubbles. By mid-semester, students are able to implement *interesting* projects involving composite graphical displays that can be modified using mouse interaction.

At all three schools, we have used OOPS for the first part of the course, as we teach the fundamentals of object-orientation: encapsulation, inheritance, and polymorphism. Early on the students use the OOPS documentation (standard Javadoc); this allows them to learn how to read and traverse realistic documentation (of reasonable size and focused scope) before using the regular API documentation, which is rather daunting.

During the second part of the course, students make the transition to Java2D and Swing, using timers for animation and (a small number of) GUI widgets for I/O. Having seen simplified versions of these things before (for example, when writing mouse methods), makes learning Java2D and Swing easier. In addition, once the students have made this transition, they can begin to read the OOPS source code, which serves to illustrate the object-oriented features we are teaching.

API Documentation (optional)

OOPS documentation and source code are both available at www.ric.edu/ksanders/cs201/.