Using Patterns to Help Students See the Power of Polymorphism

Eugene Wallingford
University of Northern Iowa
wallingf@cs.uni.edu

SIGCSE Technical Symposium
February 24, 2001

... Students have used and written classes. They have used inheritance to implement related classes. They have seen interfaces.

Polymorphism is a simple idea that is hard to master.

Students cannot master polymorphism by experiencing only simple examples, but complex examples place extra burdens on the learner.

Students with substantial procedural programming experience often require more, and more realistic, examples to overcome design inertia.

Students learn best when motivated by problems that mean something to them.

φφφ

Therefore, use common design patterns implemented in real code to illustrate the idea, use, and power of polymorphism.

The Decorator Pattern as an Illustration of Polymorphism

Decorator resonates with students because it solves a problem that they recognize and just *know* should be solvable. They see the power of polymorphism without talking about polymorphism.

Later, we use Decorator as a way to learn the `java.io` stream classes.

I do not discuss the patterns “as patterns” except in a 15-minute wrap up near the end of the semester.

The Strategy Pattern as an Illustration of Polymorphism

Strategy resonates with students because it solves a problem that they recognize and just *know* should be solvable. They see the power of polymorphism without talking about polymorphism.

Later, we use Strategy to reinforce the ideas of polymorphism and Composite objects by building compound tests.

I do not discuss patterns “as patterns” except in a 15-minute wrap up near the end of the semester.

Patterns I Use in the Same Course

Factory Method and Adaptor

State

Decorator

Composite and Null Object

Adaptor

Iterator and Factory Method

Template Method

Strategy and Composite

Problem Resolved?

Design patterns provide more complex, more realistic examples of polymorphism. Instead of talking *about* polymorphism, students **see** and **use** polymorphism.

They come to understand the power that polymorphism offers them and are then able to talk about it in a more sophisticated way.

These patterns are common tools in OO design, and so students benefit from exposure to them.

Some students will overuse the patterns "just because". The instructor must be alert to this tendency in later assignments.

A student can focus on the complexity of the design and miss the point.

Context Broadened

- This way to learn polymorphism has a long history in functional programming, even when taught in CS1.

- I have long used patterns of selection, repetition, and recursion to illustrate ideas in CS1 and CS2.

- We are now migrating these OO ideas into our first-year courses.

You can find materials from this presentation, as well as much more on elementary patterns and their use in CS instruction, at:

`http://www.cs.uni.edu/~wallingf/patterns/elementary/`

.