EE363 Fall 2012

Friday, August 31
Introduction to Java Interfaces,
and Barbara Liskov

Number Generator

- For Homework - you began with a program that generated Fibonacci Numbers...
- Let's look at that program for a minute
- What's good / what's bad?

Good Points

- Our FibonacciGenerator generates fibonacci numbers - that's it!
- Has no knowledge about how those numbers are being used.
- All user interaction is isolated away

Bad Points

- It *only* generates Fibonacci numbers...
- This wasn't a problem until I told you to also generate Prime numbers
- Now the behavior of the generator is changing!

NumberGenerator

NumberGenerator

- To solve this "problem" we *refactored* our code so that it is extensible:
  - We made a base class called NumberBehavior to describe generate method
  - We provided an implementation of that base class called FibonacciBehavior that implemented the generate method
  - We re-wrote our NumberGenerator to a behavioral interface to make it more flexible.
- Now we can add new behaviors *almost* anytime!

Where is it still too limited?

- Anyone?

What changes?

- To make the program generate Primes rather than Fibonaccis, you had to change code.
- How can we modify NumberGenerator so that we can change its behavior, without changing its code?

Liskov Substitution Principle

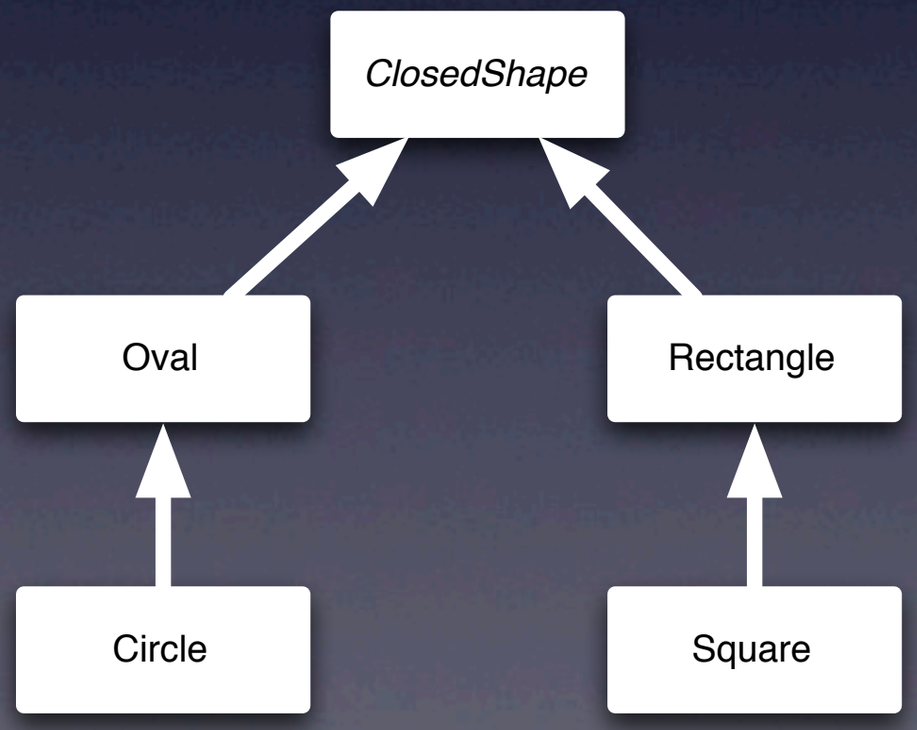
- "Subtypes must be substitutable for their base types"

  What is wanted here is something like the following substitution property: If for each object o_1 of type *S* there is an object o_2 of type *T* such that for all programs *P* defined in terms of *T*, the behavior of *P* is unchanged when o_1 is substituted for o_2 then *S* is a subtype of *T*. (Barbara Liskov, 1988)

- Behavioral sub-typing to guarantee semantic, not just syntactic, correctness

Rectangles and squares and WTF?

- A very common inheritance example shown in introductory programming courses is a Shapes hierarchy, like so:



A Closer Look...

- At first glance, this is fine -- we all know that Squares are a special case of Rectangle; and Circles are a special case of Oval -- so sub-typing seems natural to express that.
- However, Barbara Liskov would kick you right in the inheritance tree.

Liskov Violation

- Let's code up these classes, and then prove there's a Liskov violation...

- To show a Liskov violation, I have to create a program written for Rectangle, that breaks when I substitute in a Square (or visa-versa).