Objectives

- Liskov Substitution Principle
- Refactoring for Extensibility

LISKOV SUBSTITUTION PRINCIPLE

Liskov Substitution Principle (LSP)

- The substitution principle:

> 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.

- In other words...

> If a module is using a base class, then it should be able to replace the base class with a derived class *without affecting the functioning of the module.*

 Liskov & Wing, 1994

Code Smell: Using `instanceof`

```
public void drawShape( Shape shape ) {
    if ( shape instanceof Square ) {
        drawSquare(shape);
    }
    else if( shape instanceof Circle ) {
        drawCircle(shape);
    }
}
```

- Why isn't this good code?
- How could we write this in a better way?

Design by Contract

- Methods of classes declare preconditions and postconditions
  - Preconditions must be met for method to execute
  - After executing, postconditions must be true
    - Example for Rectangle's `setWidth`:
      - `myWidth == newWidth &&`
        `myHeight == oldHeight`

Design by Contract and LSP

- Methods of classes declare preconditions and postconditions
  - Preconditions must be met for method to execute
  - After executing, postconditions must be true
    - Example for Rectangle's `setWidth`:
      - `myWidth == newWidth &&`
        `myHeight == oldHeight`
- For derivatives
  - Preconditions can only be weakened
  - Postconditions can only be strengthened
  - ➥ Derivatives must adhere to constraints for base class

Design by Contract and LSP

- Recall: User interacts with interface, e.g., the base class

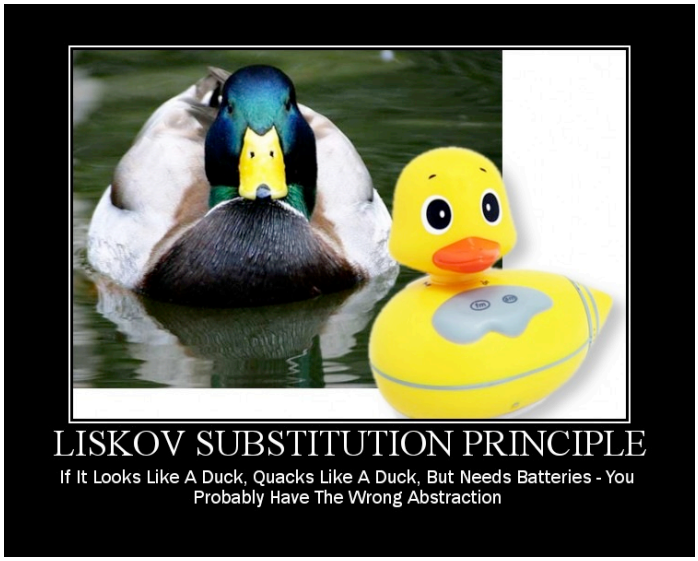
What if preconditions are stronger?
What if postconditions are weaker?

- For derivatives
  - Preconditions can only be weakened
  - Postconditions can only be strengthened
  - ➥ Derivatives must adhere to constraints for base class

Summary of LSP

- Liskov Substitution Principle (a.k.a. design by contract) is an important feature of programs that conform to the Open-Closed Principle
- Derived types must be completely substitutable for their base types
- Derived types can then be modified without consequence

http://lostechies.com/derickbailey/2009/02/11/solid-development-principles-in-motivational-pictures/



Liskov Substitution Principle (LSP)

- Named after Barbara Liskov
  - MIT Professor of Engineering
  - 2008 ACM Turing Award
  - Contributions to programming languages, pervasive computing
  - Trivia: first woman in the United States to receive a Ph.D. from a computer science department (Stanford, 1968)

Liskov & Wing, 1994

& Wing

- Jeannette Wing
  - Corporate Vice President of Microsoft Research
  - Big proponent of computational thinking as assistant director for Computer and Information Science and Engineering at the NSF from 2007 to 2010.

Discussion of Abstraction

- What does abstraction allow?
- Are there any limitations to abstraction?

Summary of Designing for Change

Use ***abstraction*** for code that is *likely to change*

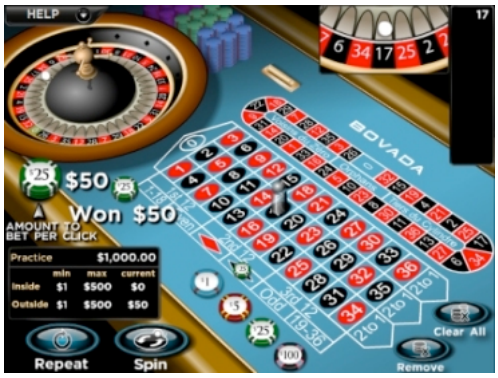
- Can depend on code that is *stable* and unlikely to change
  - Example of stable code: `System.out`

Refactoring Summary

- Write code and then *rewrite* code
  - Eye toward extensibility, flexibility, maintainability, and readability
  - Maintain correctness
- Reading/understanding other people's code can be difficult
  - Make your code readable, understandable
- Probably impossible to design/write "correctly" the first time
  - A lot harder to get the logic right, make sure you're not creating bugs, know/check the right answer…
  - Could cause yourself headaches coding this way first

REFACTORING FOR EXTENSIBILITY

Simulating a Roulette Game



In Eclipse, Import Existing Project into Workspace
`roulette.tar` on the course web site