# Façade Pattern

*CSCI-4448 - Boese*

University of Colorado **Boulder**

# Objectives

- Problem
- Definition
- Why
- Façade Examples
- How
- Comparisons

# Problem

# Problem

- Want a simplified interface since I don’t need all the features available. I only need 10 and it’s too confusing to sift through the 60 available.

# Problem

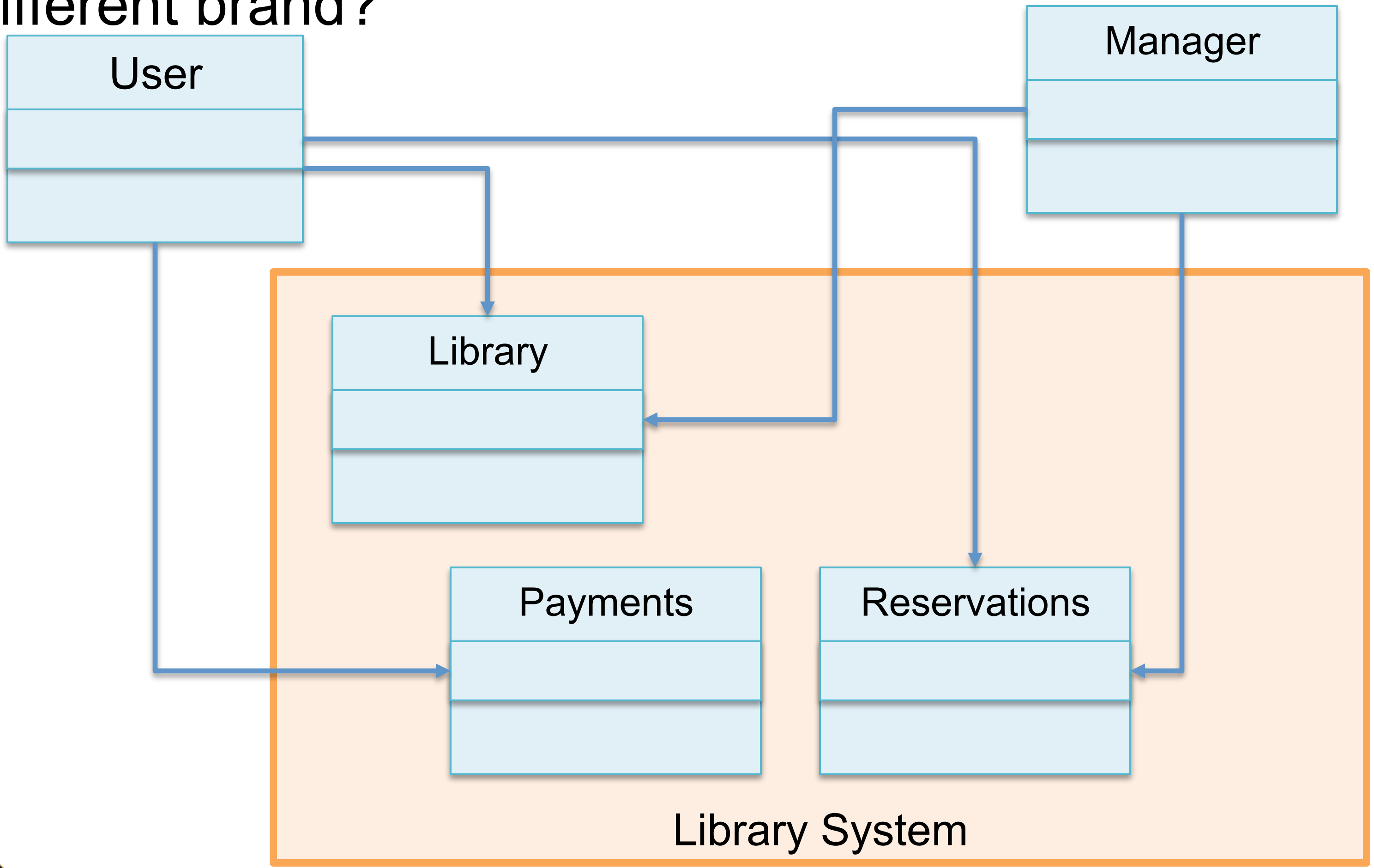
- I want to encapsulate the system and only allow a few features to be accessible to users.

# Problem

- I want to be able to draw information in a pop-up dialog box, but I don’t want to have to manipulate 10 classes to do so (Window, scrollbar, content pane, etc.). Instead I want a simple interface that sets all that up for me.

# Problem

Problem: Amount of work to replace this system with a different brand?



# Definition

# Definition

*“Provide a unified interface to a set of interfaces in a subsystem.*
*Façade defines a higher-level interface that makes the subsystem easier to use.”*

-Gang of Four

# Definition

- Name “Façade”
  - Puts a new interface (a façade) in front of original system
- Intent
  - A unified, high-level interface

# Why

**Why use Façade Pattern?**

- Do not need to use all the functionality of a complex system and can create a new class that contains all the rules for accessing that system.
  E.g., API is a simplified API than the original.
- Want to encapsulate original system.
- Want to use the functionality of the original system and want to add some new functionality as well.
- Cost of writing this new class is less than the cost of
  - Everybody learning how to use the original system
  - Maintenance in the future

# Subsets

# Subset

- Subset of the full set of functionality
  - Simplify an otherwise complex programming task
    - ***Easier to use***
    - ***More readable***

# Subsets

## Math Functions Available

acos(double x) -- Compute arc cosine of x.
asin(double x) -- Compute arc sine of x.
atan(double x) -- Compute arc tangent of x.
atan2(double y, double x) -- Compute arc tangent of y/x, using the s
arguments to determine the quadrant of the return value.
ceil(double x) -- Get smallest integral value that exceeds x.
cos(double x) -- Compute cosine of angle in radians.
cosh(double x) -- Compute the hyperbolic cosine of x.
div_t div(int number, int denom) -- Divide one integer by another.
exp(double x -- Compute exponential of x
fabs (double x ) -- Compute absolute value of x.
floor(double x) -- Get largest integral value less than x.
fmod(double x, double y) -- Divide x by y with integral quotient and r
frexp(double x, int *expptr) -- Breaks down x into mantissa and expo
labs(long n) -- Find absolute value of long integer n.
ldexp(double x, int exp) -- Reconstructs x out of mantissa and expo
ldiv_t ldiv(long number, long denom) -- Divide one long integer by a
log(double x) -- Compute log(x).
log10 (double x ) -- Compute log to the base 10 of x.
modf(double x, double *intptr) -- Breaks x into fractional and integer
pow (double x, double y) -- Compute x raised to the power y.
sin(double x) -- Compute sine of angle in radians.
sinh(double x) - Compute the hyperbolic sine of x.
sqrt(double x) -- Compute the square root of x.
srand(unsigned seed) -- Set a new seed for the random number generator (rand).
tan(double x) -- Compute tangent of angle in radians.
tanh(double x) -- Compute the hyperbolic tangent of x.

## Math Functions that CSCI-1300 need to know

ceil(double x) -- Get smallest integral value that exceeds x.

exp(double x -- Compute exponential of x

abs (double x ) -- Compute absolute value of x.

floor(double x) -- Get largest integral value less than x.

pow (double x, double y) -- Compute x raised to the power y.

sqrt(double x) -- Compute the square root of x.

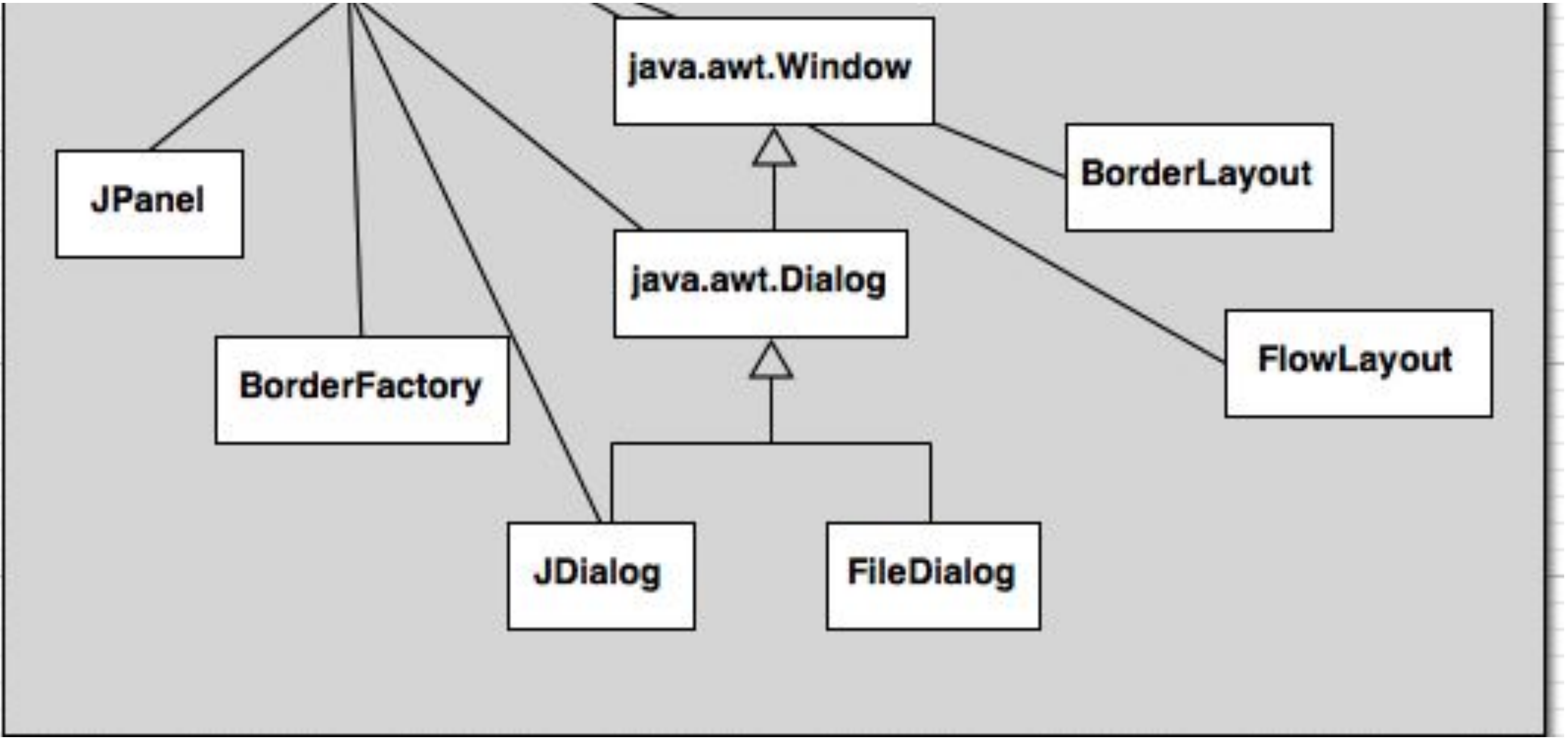
srand(unsigned seed) -- Set a new seed for the random number generator (rand).

# Simplification

# Problem

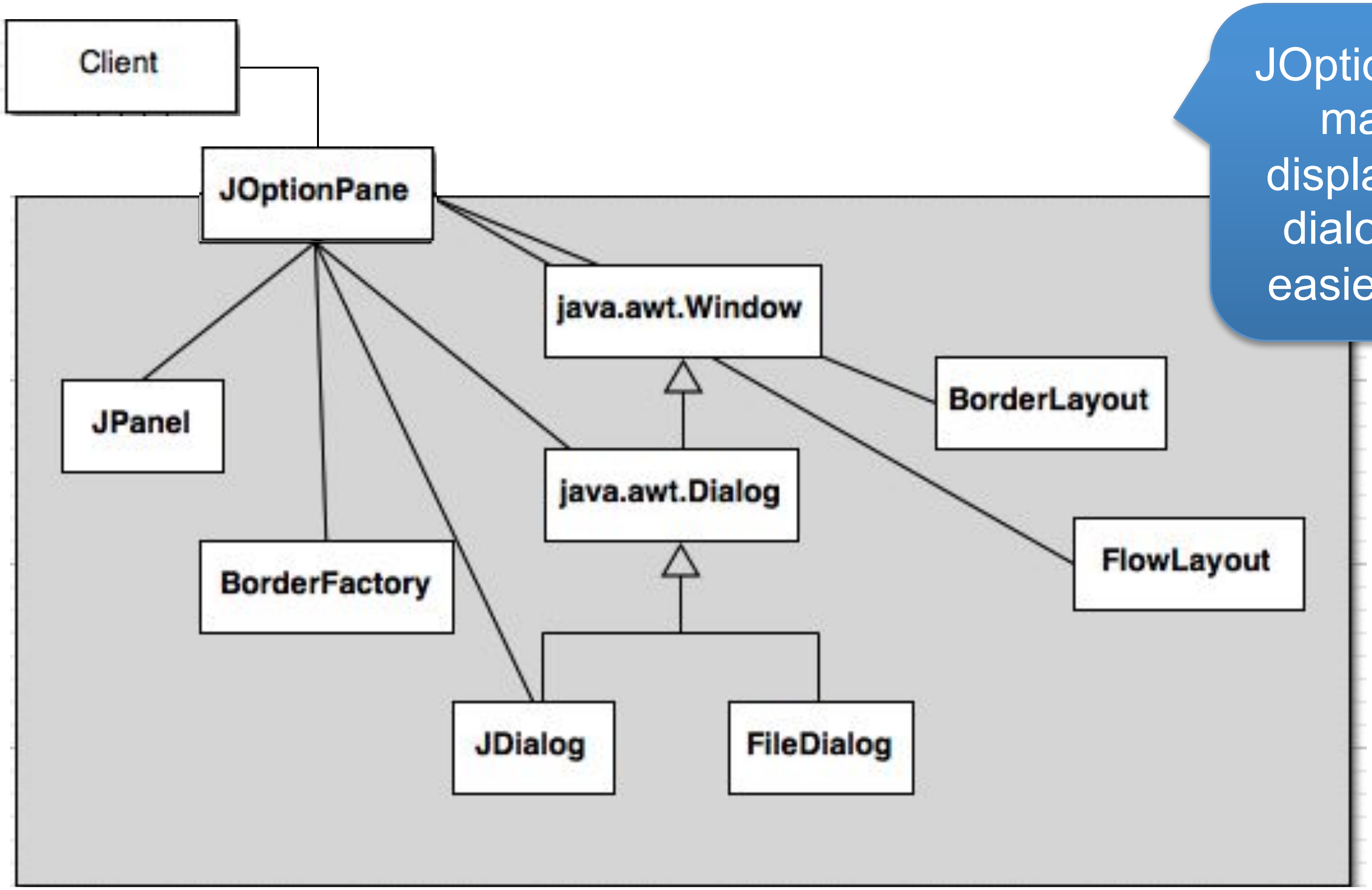
- To create a good dialog box in Java requires
  - JDialog
  - BorderFactory
  - Jpanel
  - Layout
  - Window

What a pain to use!



# Subsets: Simplify: Easier to Use

Example: Java Applets: JOptionPane is a Façade Pattern for a Dialog Box



JOptionPane makes displaying a dialog box easier to do

# Subsets: Simplify – Easier to Read

Example: Java Applet’s GridBagLayout

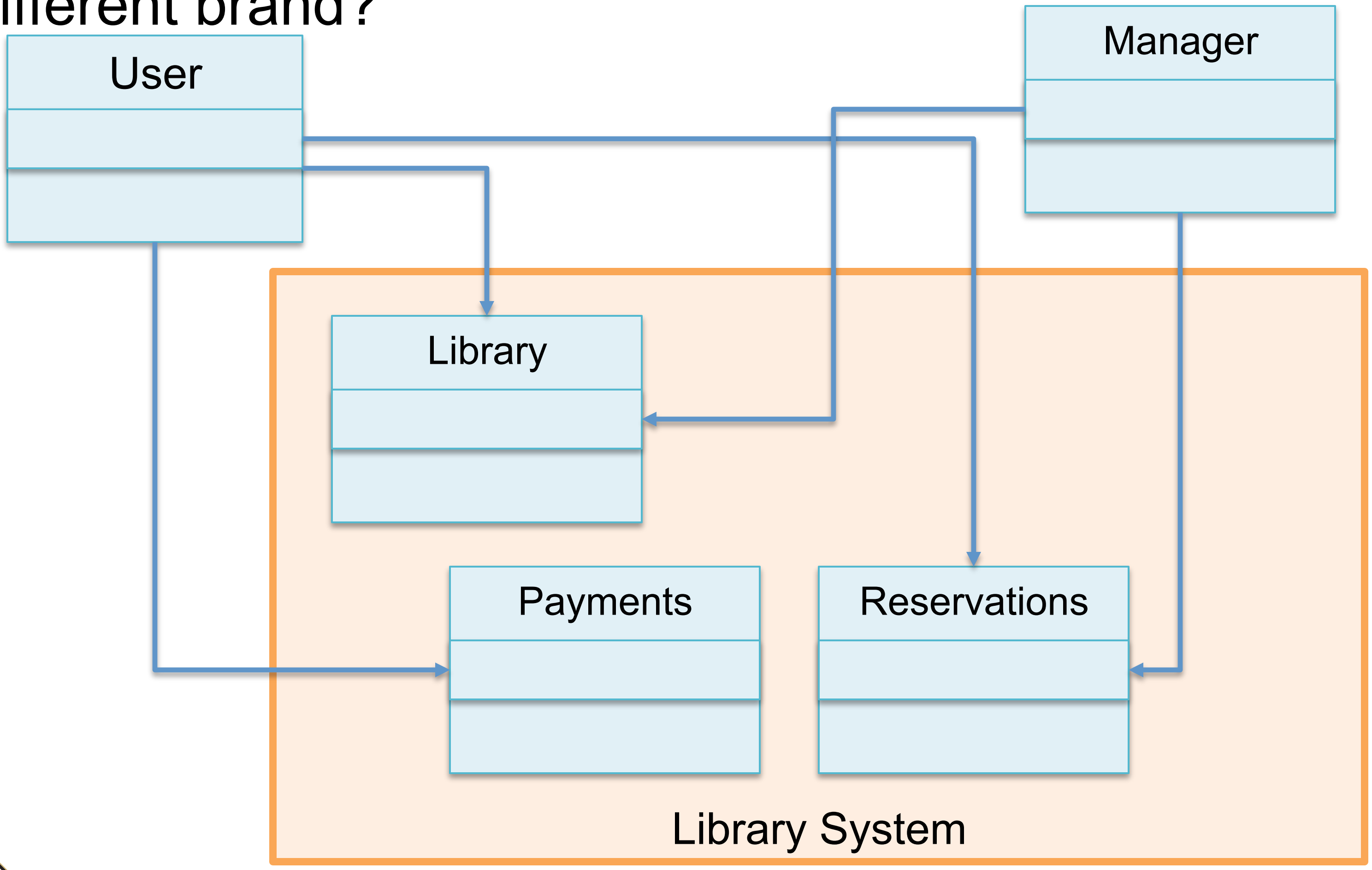
```
GridBagConstraints c = new GridBagConstraints();
c.weightx = 1;
c.weighty = 1;
c.gridx = 0;
c.gridy = 0;
c.anchor = GridBagConstraints.SOUTHWEST;
c.fill = GridBagConstraints.BOTH;
gridbag.setConstraints(B1,c);
```

```
GridBagConstraints c =
             new MyGrid(1,1,0,0, MyGrid.SW, MyGrid.BOTH);
Gridbag.setConstraints(B1,c);
```

# Abstraction for Swapping Out Systems

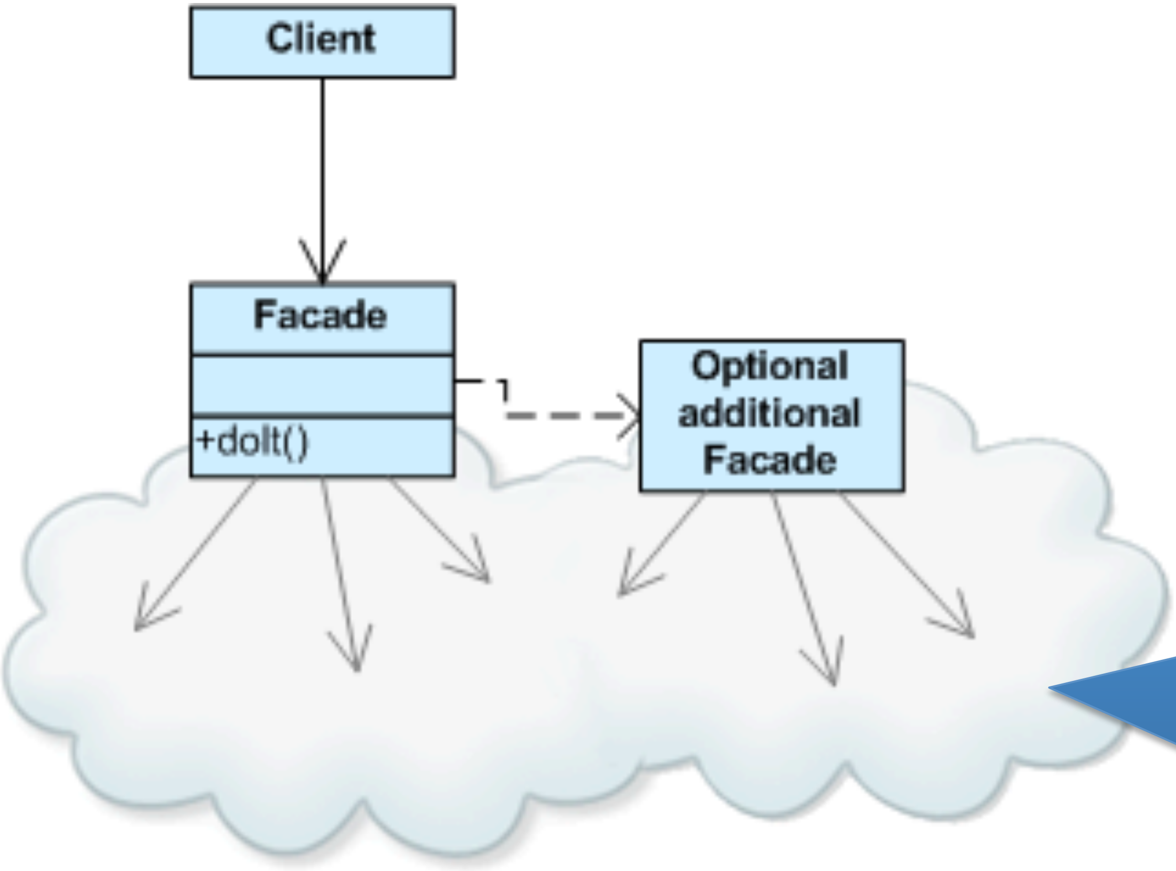
# Subsets: Abstraction

Problem: Amount of work to replace this system with a different brand?



# Subsets: Abstraction

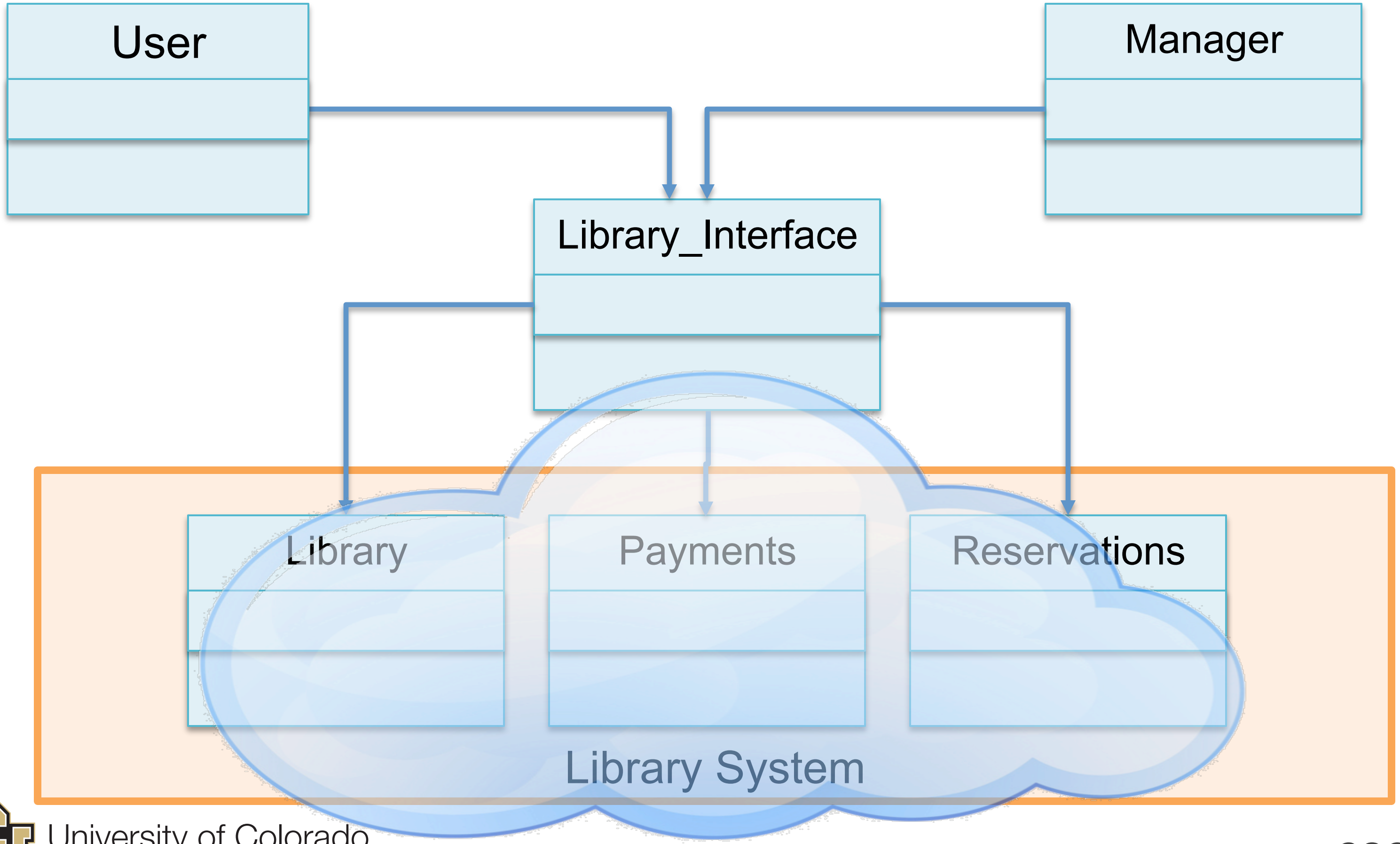
Subset: Abstraction for swapping out systems



If we change the underlying system used, we only need to modify the Façade classes.

# Subsets: Abstraction

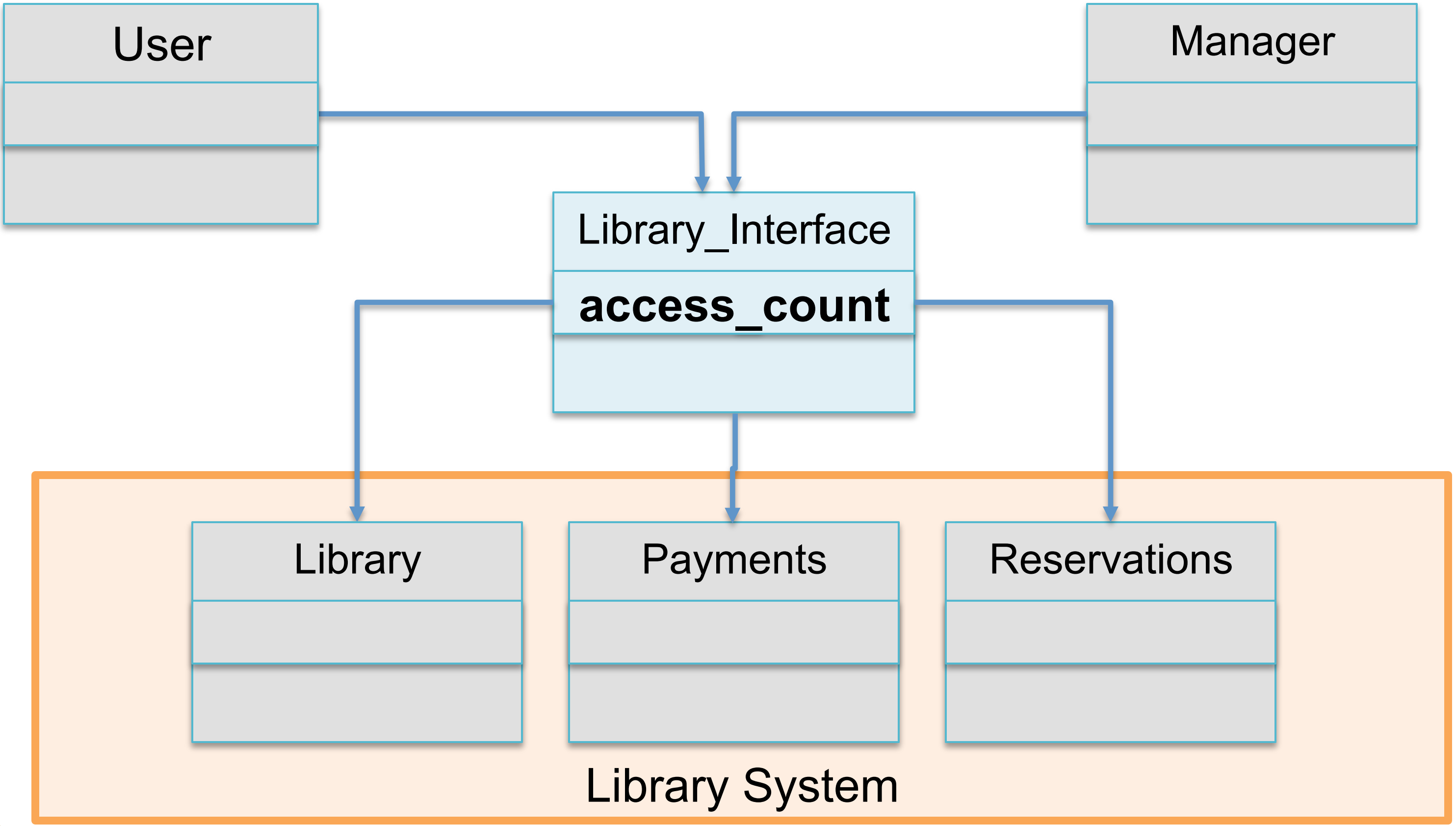
Solution: Replace Library System and only need to change the Library_Interface



# Track Usage

# Subsets: Abstraction

- Problem: Amount of work to replace this system with a different brand?



# How

# How

- Identify a simpler, unified interface for the subsystem or component.
- Create a class(es) that call the methods/functions in the core system (wrapper).
- Client calls methods/functions in the classes you create.
- If the system is replaced, the classes you created will need to be modified to handle calling the new system. However, the client calls to your class should remain the same.
    - The client uses (is coupled to) the Facade only.

# Comparisons

# Comparisons

- Facade defines a new interface, whereas Adapter uses an old interface
- Adapter and Facade are both wrappers; but they are different kinds of wrappers. The intent of Facade is to produce a simpler interface, and the intent of Adapter is to design to an existing interface. While Facade routinely wraps multiple objects and Adapter wraps a single object; Facade could front-end a single complex object and Adapter could wrap several legacy objects.
- Flyweight shows how to make lots of little objects, whereas Facade shows how to make a single object represent an entire subsystem.
- Facade objects are often Singletons because only one Facade object is required.