CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2014
Doc 19 Facade & Mediator
April 24, 2014

Facade

Size

Item	Source Lines of Code (Millions)
F-22 Raptor US jet fighter	1.7
Boeing 787	6.5
S-class Mercedes-Benz radio & navigation system	20
Mac OS 10.4	86
Premium class automobile	~100
Debian 4.0	283

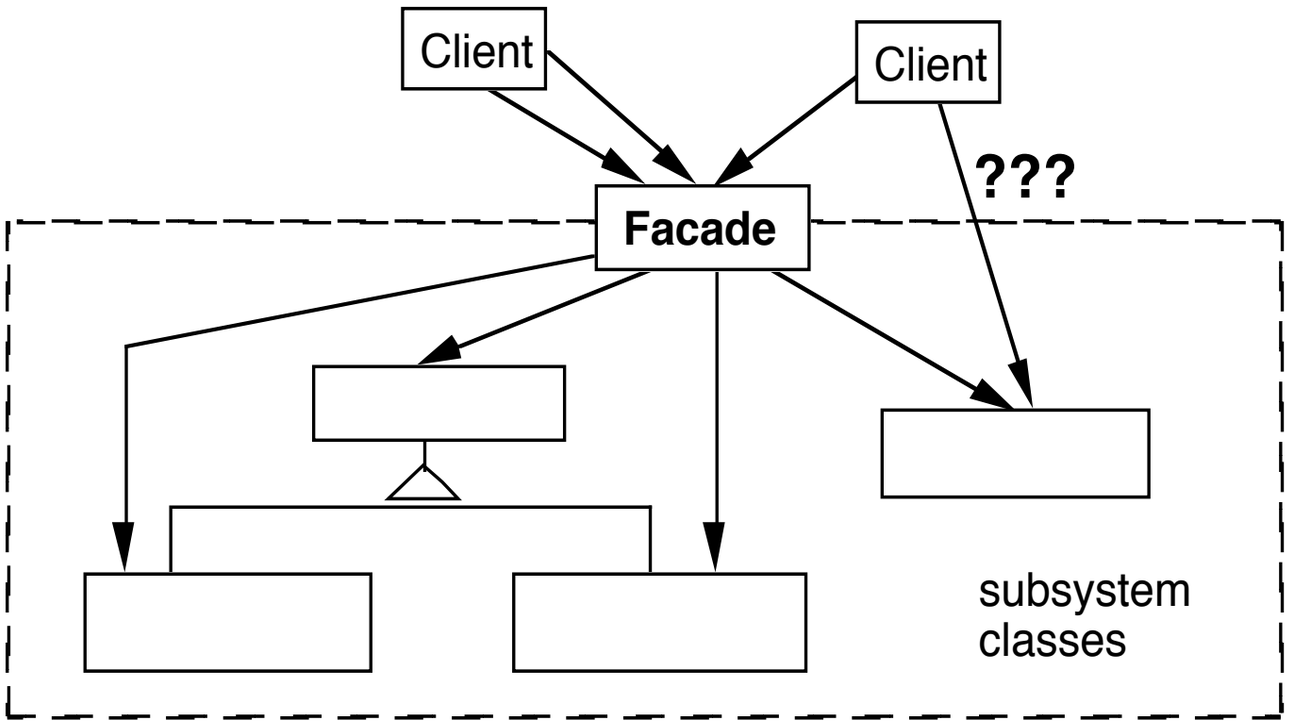
Design Patterns text contains under 8,000 lines

Reference: This Car Runs on Code, http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code
Source lInes of code, http://en.wikipedia.org/wiki/Source_lines_of_code

The Facade Pattern

Create a class that is the interface to the subsystem

Clients interface with the Facade class to deal with the subsystem



Consequences of Facade Pattern

It hides the implementation of the subsystem from clients

It promotes weak coupling between the subsystems and its clients

It does not prevent clients from using subsystem classes directly, should it?

Facade does not add new functionality to the subsystem

Public versus Private Subsystem classes

Some classes of a subsystem are

- public
  - facade
- private

Compiler Example

The VisualWorks Smalltalk compiler system has 75 classes

Programmers only use Compiler, which uses the other classes

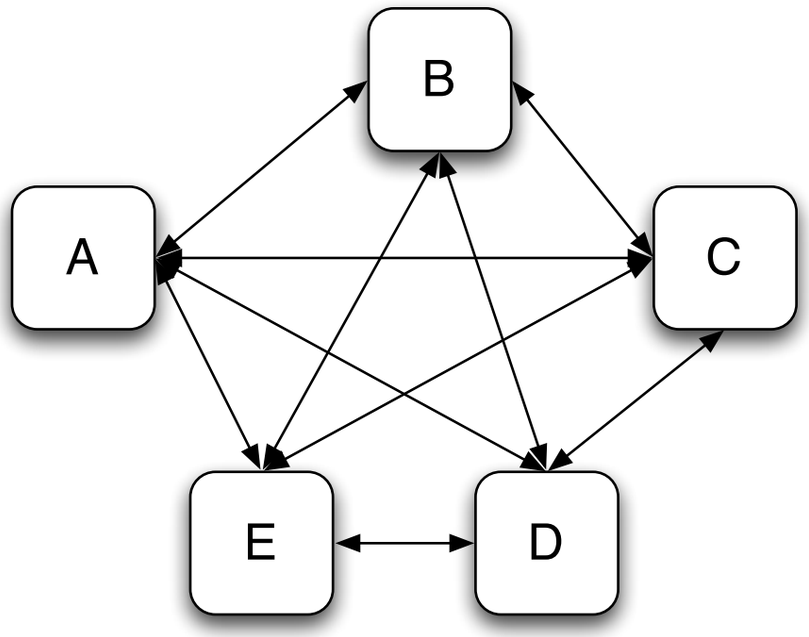
Compiler evaluate: '100 factorial'

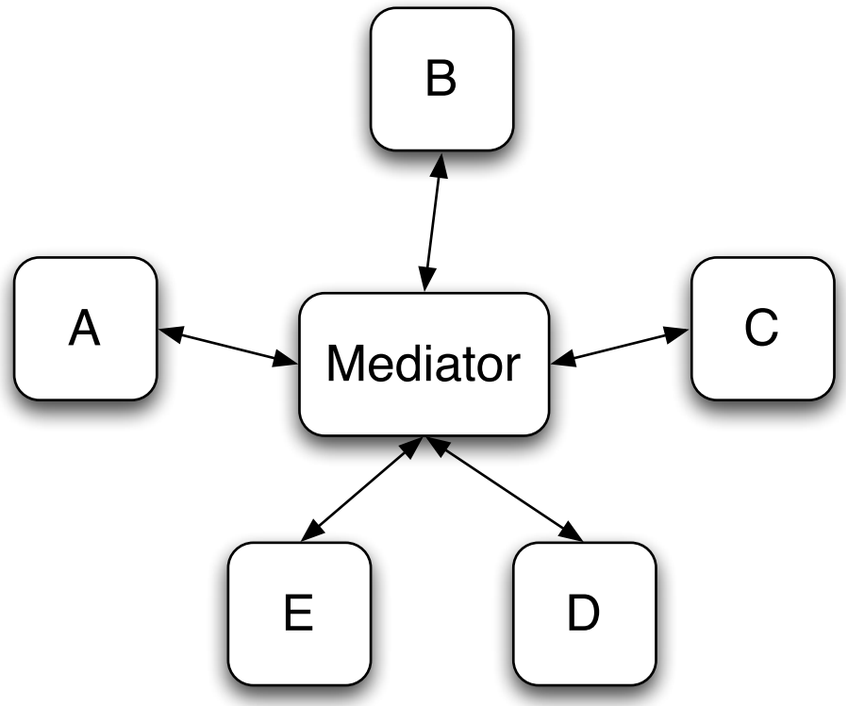
```
| method compiler |
method := 'reset
        "Resets the counter to zero"
        count := 0.'.

compiler := Compiler new.
compiler
        parse:method
        in: Counter
        notifying: nil
```

Mediator

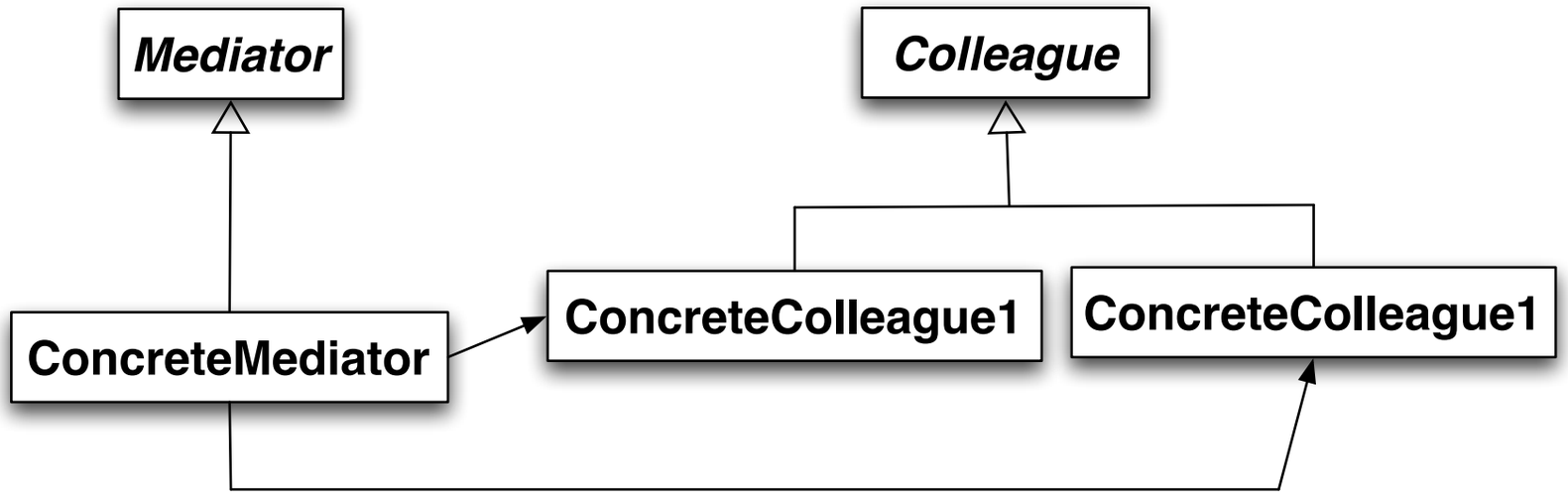
Mediator

A mediator controls and coordinates the interactions of a group of objects





Structure


Mediator
Colleague
ConcreteMediator
ConcreteColleague1
ConcreteColleague1

Participants

Mediator

Defines an interface for communicating with Colleague objects

ConcreteMediator

Implements cooperative behavior by coordinating Colleague objects

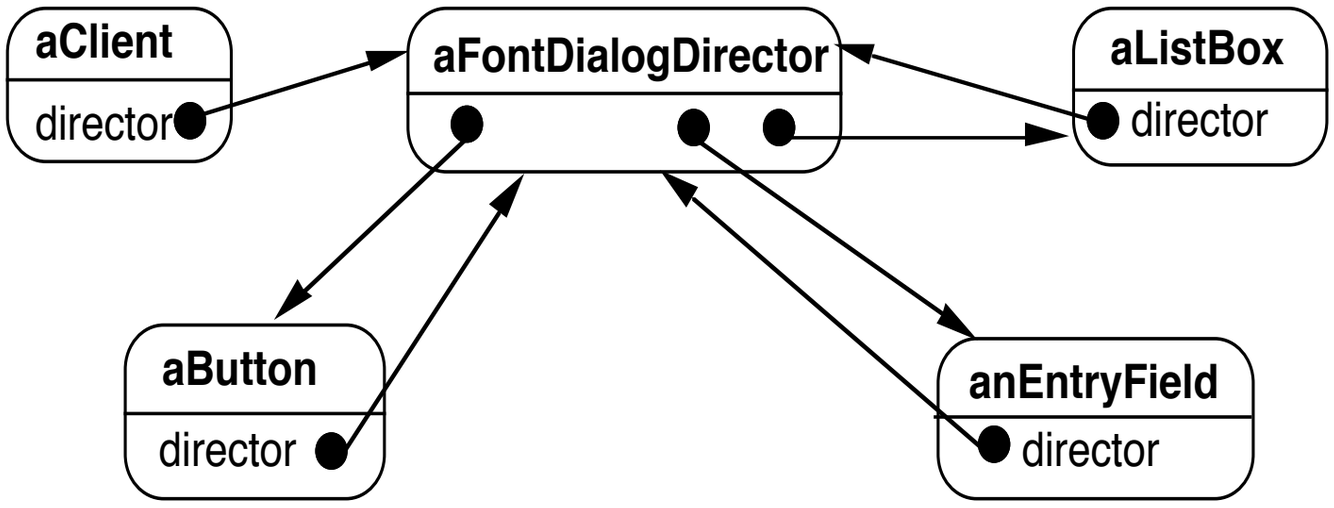
Knows and maintains its colleagues

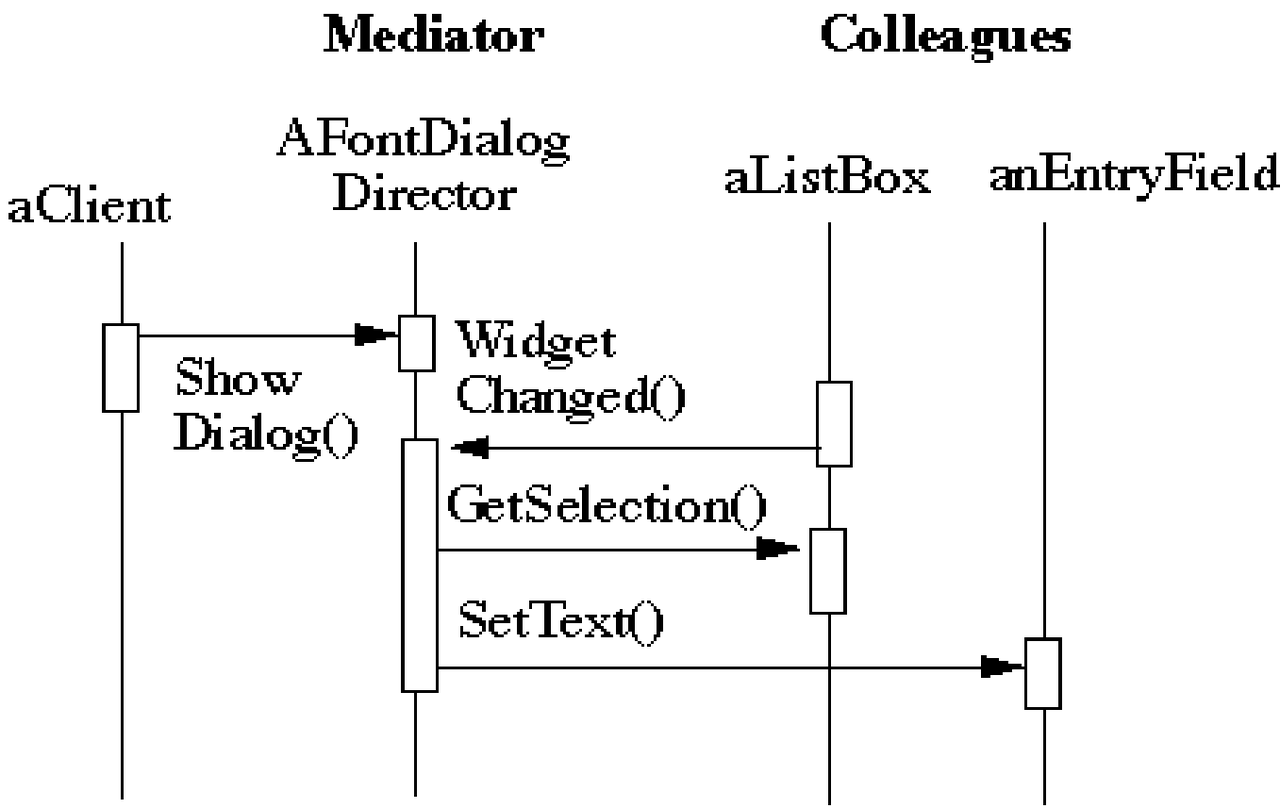
Colleague classes

Each Colleague class knows its Mediator object

Each colleague communicates with its mediator whenever it would have otherwise communicated with another colleague

Motivating Example - Dialog Boxes





How does this differ from a God Class?

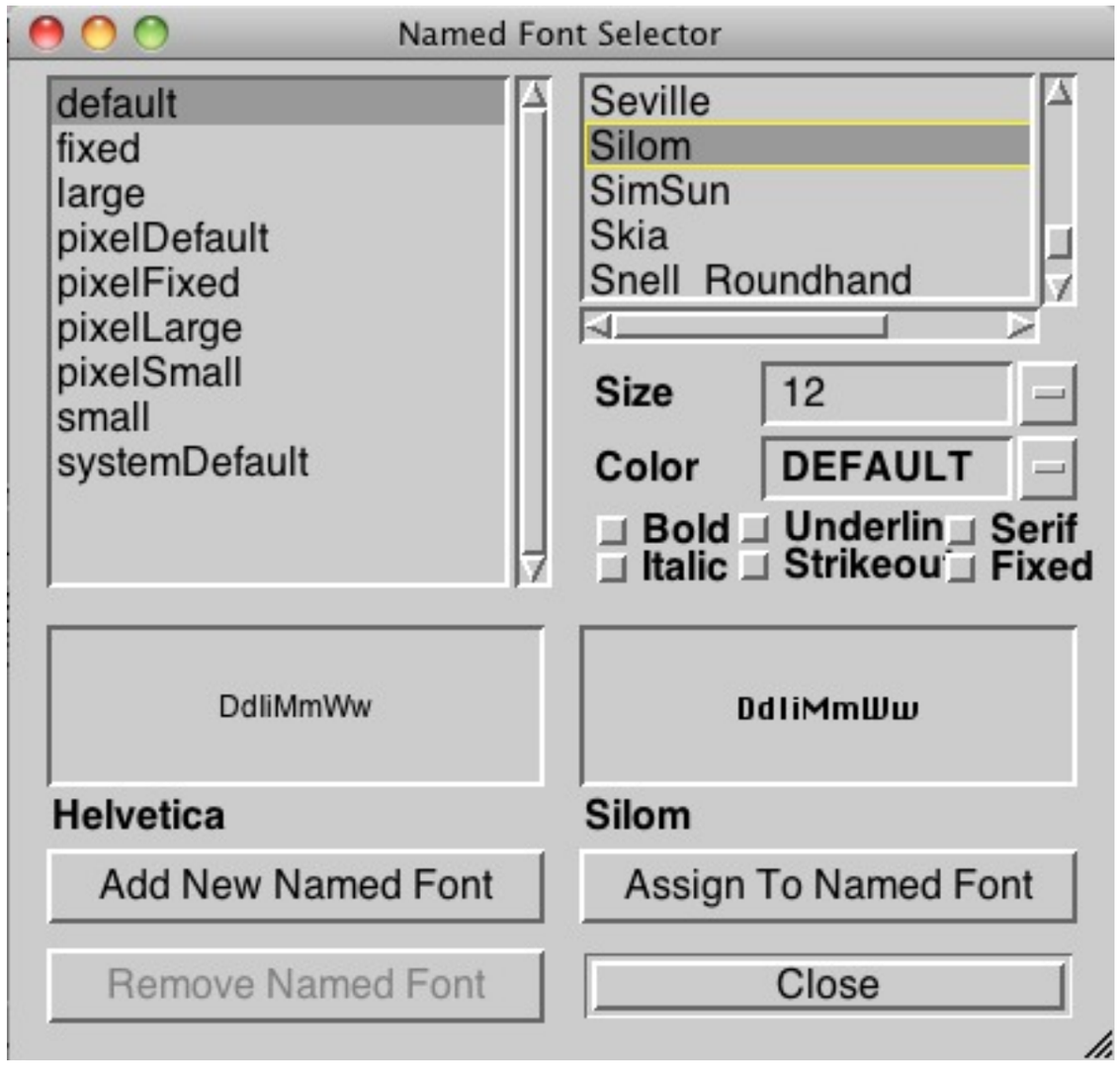
When to use the Mediator Pattern

When a set of objects communicate in a well-defined but complex ways

When reusing an object is difficult because it refers to and communicates with many other objects

When a behavior that's distributed between several classes should be customizable without a lot of subclassing

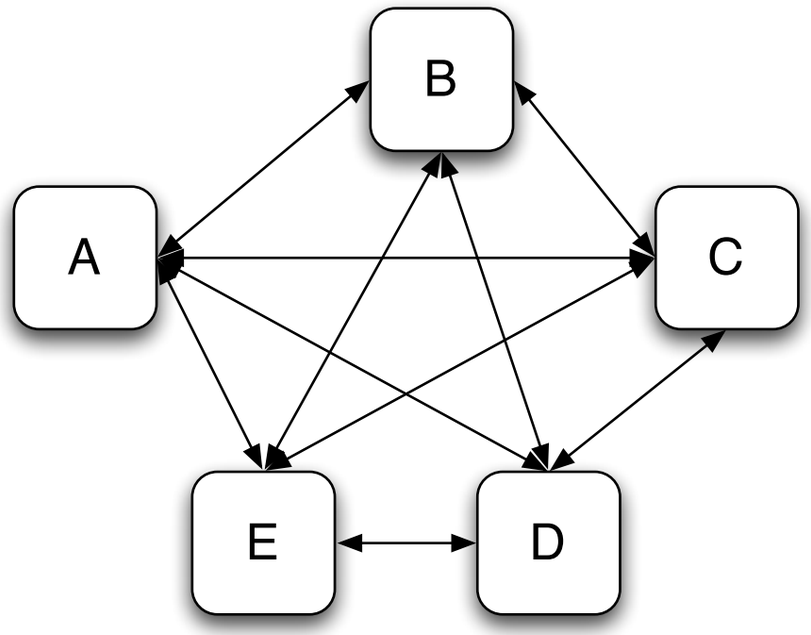
Classic Mediator Example



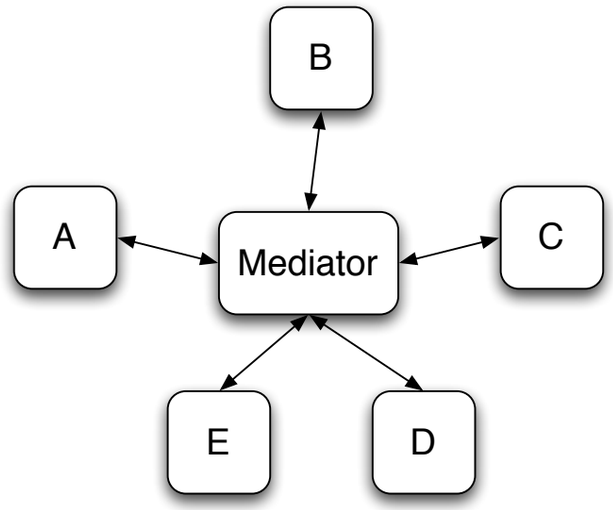
Simpler Example

Non Mediator Solution

```
class OKButton extends Button {
      TextField password;
      TextField username;
      Database userData;
      Model application;

      protected void processEvent(AWTEvent e) {
            if (!e.isButtonPressed()) return;
            e.consume();
            if (password.getText() = "") {
                  notifyUser("Must enter password");
                  return;
            }
            if (username.getText() = "") {
                  notifyUser("Must enter user name");
                  return;
            }
            if (!userData.validUser(password.getText(), username.getTest()))
                  notifyUser("Invalid username & password");
                  return;
            }
```



Mediator Solution




```
class LoginDialog extends Panel {
      TextField password;
      TextField username;
      Database userData;
      Button ok, cancel;

      protected void actionPerformed(ActionEvent e)  {
            if (!e.isButtonPressed() or e.getSource() != ok) return;
            if (password.getText() = "") {
                  notifyUser("Must enter password");
                  return;
            }
            if (username.getText() = "") {
                  notifyUser("Must enter user name");
                  return;
            }
            if (!userData.validUser(password.getText(), username.getTest()))
                  notifyUser("Invalid username & password");
                  return;
            }
```

What is Different?

Non Mediator Example

Special Button class
OK button coupled to text fields

Mediator Example

No specialButton class
LoginDialog coupled to text fields

Logic moved from button class to LoginDialog

But

Java's event mechanism promotes mediator solution