Design Patterns

Chicken + Egg

do design
understand design patterns

CVS permission issues:
Contact LACY@CEC.WUSTL.EDU

What is a design pattern?
Common relationship among objects or classes.
Abstraction of abstractions!!
Quiz 1: Name a design pattern & define it or give a brief example.

Put your name & lab section at the top!!
Problems:

1. User changes their mind about format.
   - Multi choice $\rightarrow$ Short Answer
   - New question object, but other questions refer to it

2. All combinations of question types + format types

Want: independently choose among 2 class hierarchies

Solution: BRIDGE PATTERN
Problem: How to manage survey navigation

Plan A: Survey class has logic to choose (custom)

But: Hard to extend package to change the logic

Plan B: Plug-in functionality "decider" object

But: A bit ad-hoc

Solution: Use the Iterator Pattern

hasNext() — end of survey then return false

next() — return the next question

As far as the rest of the impl. can tell, just iterating over the survey.
Problem: Nested surveys?  (Sections, subsections...)

Solution: Composite Pattern

Extreme case:
Problem: Have a composite of nested survey items.

Want: Multiple ways to save the survey to a file. (Lab 2)

Possibilities:

Plan A: Each object (survey, question, choice...) knows how to "print" itself for each output type.

Bunch of methods like printHTML, etc.

... call recursively.

MESS! — sprinkled all over the code

Solution (next page)
Solution: **Visitor Pattern**

Each object in the composite (survey, question, choice...) has a method:

```java
void visit(Visitor v) {
    v.visit(this);
    for (all my children c)
        c.visit(v)
}
```

in visitor:

```java
visit(Survey s) {
    visit(Question q) {
        visit(Choice c) {
        }
    }
}
```

1. Create visitor
2. `survey.visit(visitor)`
Problem: Different views for different situations

Possibilities: Plan A: Could have each item have methods like:
question, create UserView()
question, create EditorView()

Mess! — No separation of concerns
Better... new UserView(Survey)

Even better...
Factory Pattern ...

ViewFactory
View createView(Survey)

UserViewFactory EditorViewFactory PlaybackFactory
Problem: Navigating through survey & (back & forth)
Display a question, but might already have been there
Don't want: New view every time.

Solution: Flyweight pattern

```
getView(Question q)
if don't already have a view for q
create one
return the view for q
```

Might want singleton pattern with this approach. (One cache)
**Problem:** Separate what user sees from the logic that handles user interaction.

- Model
- Controller
- View

**Model** ➔ **Controller** ➔ **View**

- Property change
- Mutating method call
- Events
- Update

**Publish/Subscribe (Observer) pattern**

- Controller registers itself with the View
- View notifies controller about events