OOAD

"There is no silver bullet."
—Fred P. Brooks, Jr.

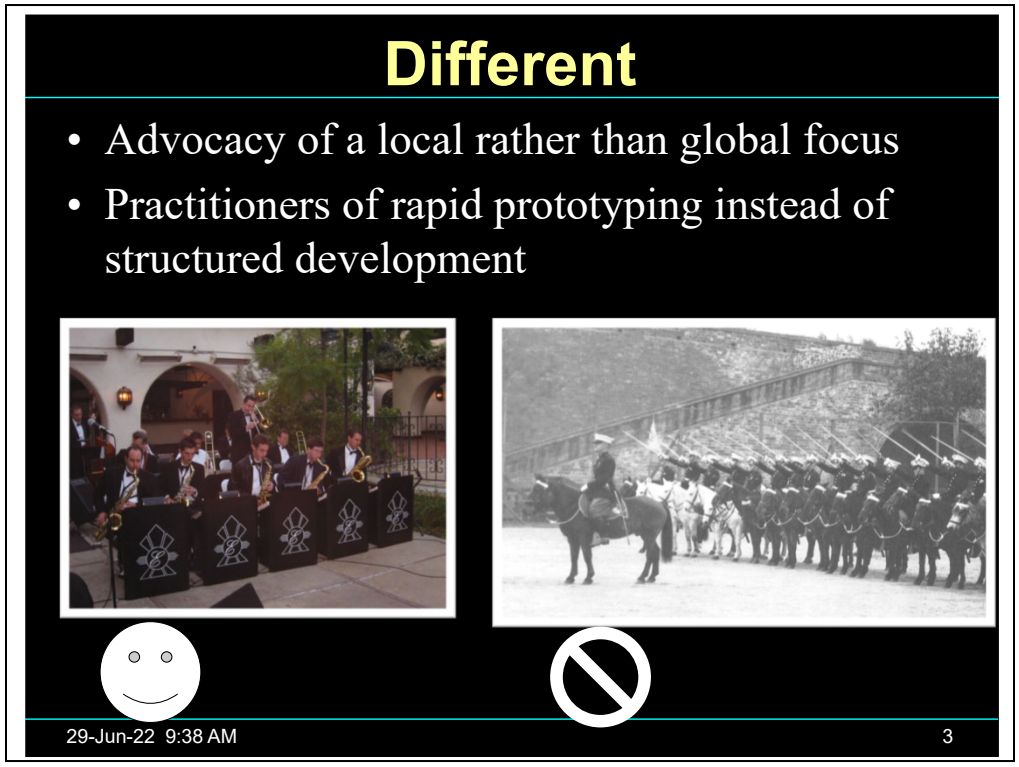
OO Design is an Art

29-Jun-22 9:38 AM 2

OOAD books and classes can only enhance the innate talents of individuals and make them the best OO engineers they can be.

The education of artists is not focused on technique, process or tools.

> The majority of an art education combines ideas, history, appreciation, experience and constructive criticism.

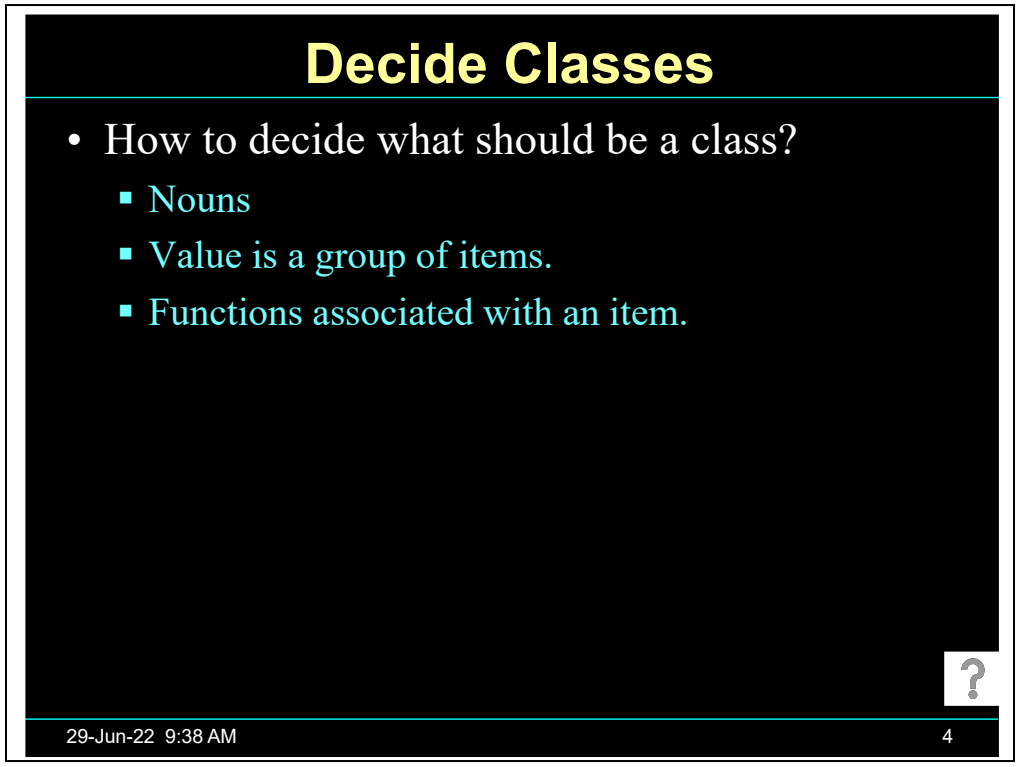


Collaborative rather than imperial management style

OOD -

Commitment to design based on coordination and cooperation rather than control

Driven by internal capabilities instead of conforming to external procedures.



Q022: telLocalNumber

Q01: IsSameString

Class should be highly cohesive

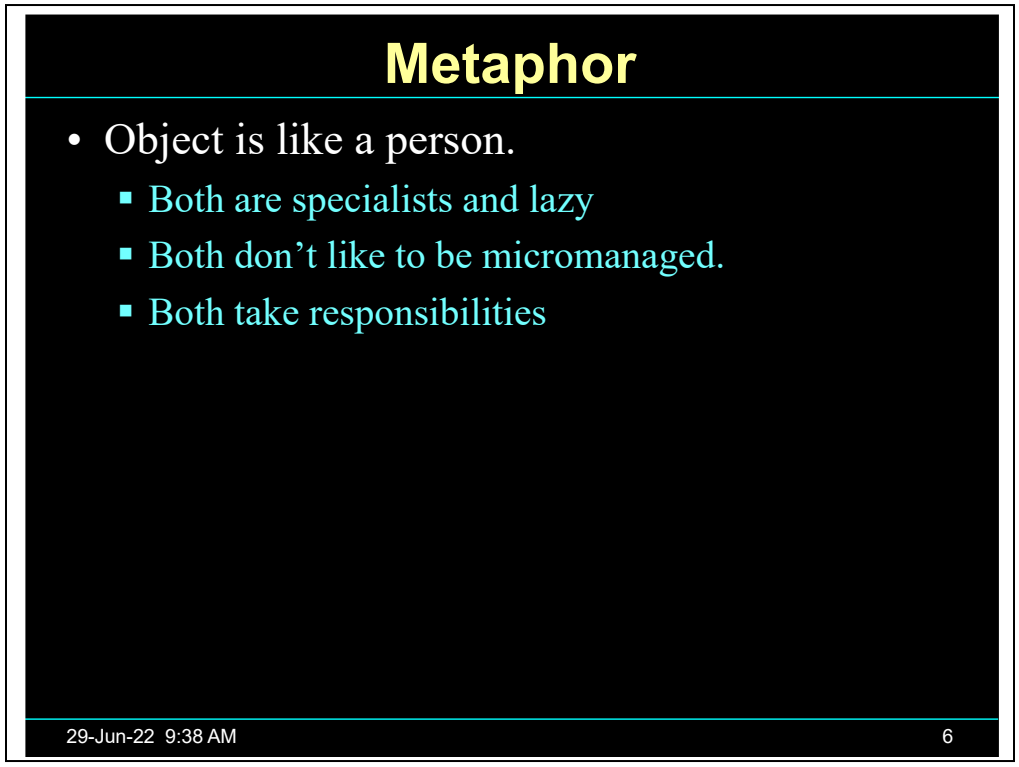
A Class should have a single well focused purpose.

A Class should do one thing and do it well.

Find the classes

- Selling soft drinks on a vending machine.
  - Software will control the functions of the vending machine.
  - First the user enters some money. The machine displays the money entered so far. The products that can be bought light up. The user chooses his option. The vending machine dispenses the product and the change.

Answer: VendingMachine, MoneyBox, Screen, PriceList, SoftDrink, SoftDrinkList, SoftDrinkDispenser, Safe

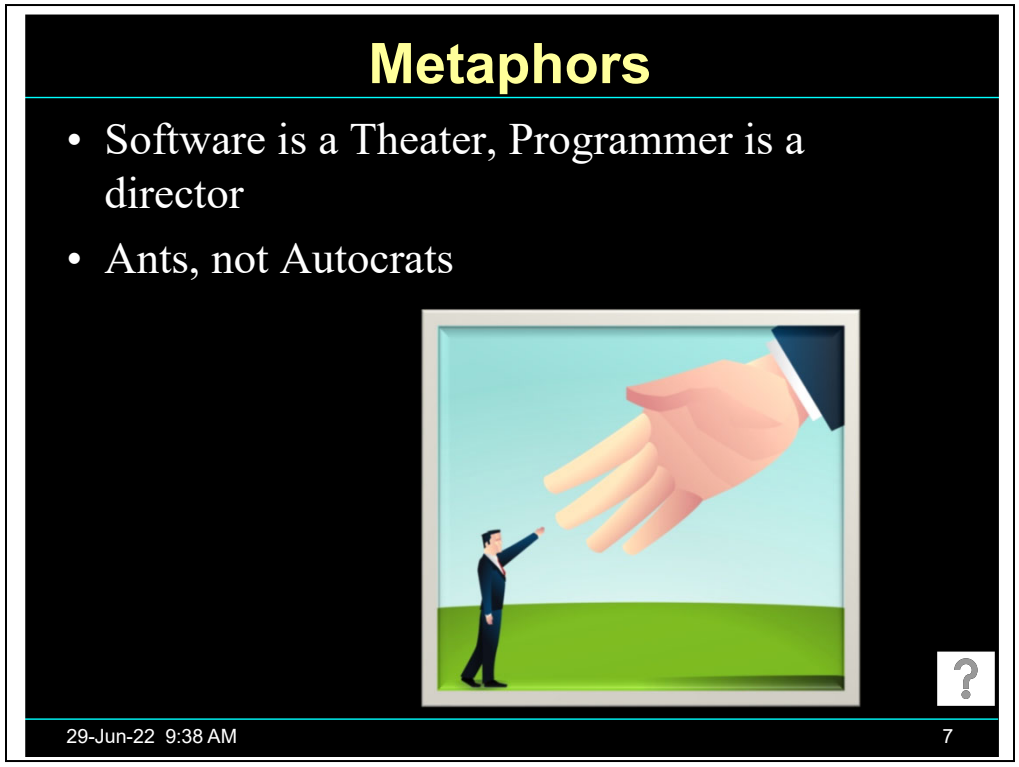


Distributed cooperation and communication must replace hierarchical centralized control as an organizational paradigm. E.g. A traffic signal and cars.

Like people, software objects are specialists and lazy. A consequence of both these facts is the distribution of work across a group of objects. Take the job of adding a sentence to a page in a book. Granted, it might be quite proper to ask the book, "Please replace the sentence on page 58 with the following." (The book object is kind of a spokesperson for all the objects that make up the book.) It would be quite improper, however, to expect the book itself to do the work assigned. If the book were to do that kind of work, it would have to know everything relevant about each page and page type that it might contain and how making a simple change might alter the appearance and the abilities of the page object. Plus the page might be offended if the book attempted to meddle with its internals.

The task is too hard (lazy object) and not the book's job (specialist object), so it delegates—merely passes to the page object named #58 the requested change. It's the page object's responsibility to carry out the task. And it too might delegate any part of it that is hard—to a string object perhaps.

Objects, like the people we metaphorically equate them to, can work independently and concurrently on large-scale tasks, requiring only general coordination. When we ask an object collective to perform a task, it's important that we avoid micromanagement by imposing explicit control structures on those objects. You don't like to work for a boss who doesn't trust you and allow you to do your job, so why should your software objects put up with similar abuse?



Q57srp – Students Q58srp – Servers

Hierarchical and centralized control is the anathema in OO. Complex systems are characterized by simple elements, acting on local knowledge with local rules, give rise to complicated patterned behavior. Object can inherit like a person.

For example, suppose an airplane object has a responsibility to report its location. This is a hard task because the location is constantly changing; a location is a composite structure (latitude, longitude, altitude, direction, speed, and vector); the values of each part of that structure come from a different source; and someone has to remember who asked for the location and make sure it gets back to them in a timely fashion. If the task is broken up so that

- The airplane actually returns a location object to whoever asked for it after appending its ID to the location so that there is no confusion about who is where. (We cannot assume that our airplane is the only one reporting its location at any one time.)

- An instrument cluster keeps track of the instruments that must be asked for their current values and knows how to ask each one in turn for its value (a collection iterating across its contents).

- An instrument merely reports its current value.
- A location object collects and returns a set of label:value pairs (altitude:15,000 ft.).

None of the objects do anything particularly difficult, and yet collectively they solve a complicated problem that would be very hard for any one of them to accomplish individually.

Guidelines

- DRY

29-Jun-22 9:38 AM 8

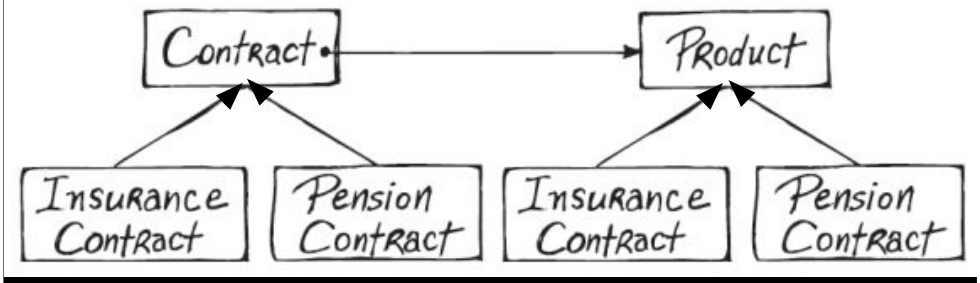
Q021: BookRentals; Q11: BookRental; Q33 – SurveyData; Q37 – FOC, TT

Q10 – Jbutton

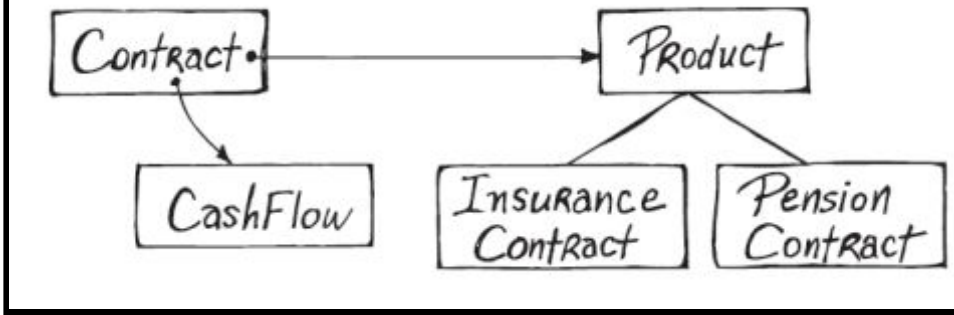
Q34 – replace; Q35 – BookRental

There are three numbers in software: 0, 1, and infinity. 0 represents the things we do not do in a system (we do those for free). 1 represents the things we do once and only once. But at the moment we do something twice, we should treat it as infinitely many and create cohesive services that allow it to be reused.

Improve

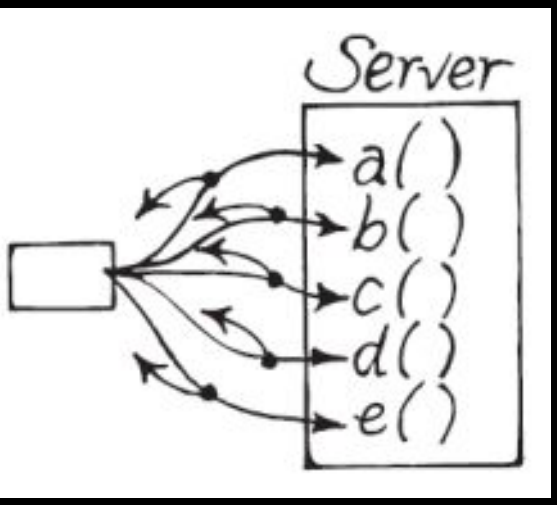

Contract
Product
Insurance Contract
Pension Contract
Insurance Contract
Pension Contract

Eliminate Parallel Hierarchies



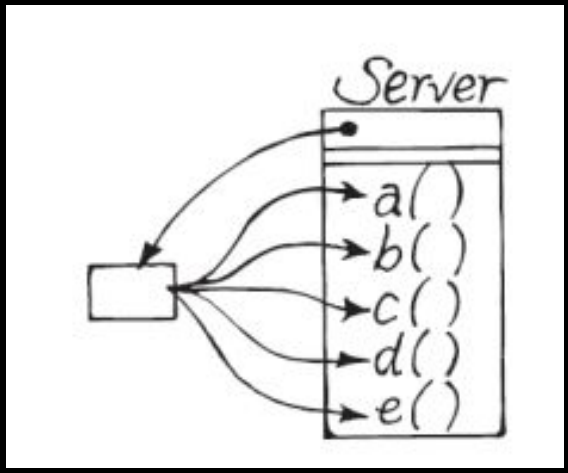
Reduce Coupling - Problem

```
Server s = new Server();
s.a(this);
s.b(this);
s.c(this);
s.d(this);
s.e(this);
```



Reduce Coupling - Solution

```
Server s = new Server(this);
s.a();
s.b();
s.c();
s.d();
s.e();
```



Better still: use Observer design pattern