Object-Oriented Programming

CS420/520 Winter 2012

Prof Andrew P. Black

Administrivia

Instructor:

- Andrew Black,
  - 503 725 2411 (office) 503 803 1669 (cell).
  - email: *black@cs.pdx.edu*, put [CS520] in subject header
  - Office (FAB 115-10): Monday 11:30–noon and Wednesday 11:15–11:50, or by appointment. Feel free to telephone for an appointment.
- Class meets: Tuesday and Thursday 14:00–15:50 in FAB 040-08

Web Resources

http://web.cecs.pdx.edu/~black/OOP/

Portland State UNIVERSITY
Department of Computer Science

CS 420/520 Object-Oriented Programming

Sign up for the class mailing list!

Winter Quarter 2013

Tuesday and Thursday 14:00–15:50 in FAB 040-08
CRN 41019 (CS 420) Restricted to - MCECS; JR SR PB; majoring in BMI, CMPE, CS or EE
If you would like me to waive the restrictions, please come and see me.
CRN 41034 (CS520)

Instructor
Andrew Black
email: black
telephone: 503 725 2411

Prof Black will hold office hours in FAB 115-10 Monday 11:30–noon and Thursday 10:30–11:50. It's also fine to just wander by his office and see if he's free, or send an email or telephone to set up an appointment at an alternative time.

Grader
TBA
email:

Prerequisites

No familiarity with object-oriented programming is assumed; students who have experience with Java or C++ should be prepared to unlearn some of what they think they know. Prior experience with Smalltalk is not required. The class assumes a working knowledge of discrete mathematics and machine organization — the fundamentals of instruction coding and data representation in the computer's memory. (Students who have completed PSU CS 310 and CS 340 will be well prepared)

Textbooks

Educational Goals

To change the way that you think

- Learn a new set of concepts and the vocabulary to talk about them:
  - object, class, message, protocol, instance variable, method, refactoring
  - delegation, genericity, traits, incremental programming, type-checking, polymorphism, encapsulation, and set-based abstraction

To teach new skills

- Learn to program in an object-oriented languages with a non-conventional structure: Smalltalk
- A preview of a new language: Grace
- Learn a new methodology: test-driven development, which combines design & programming.
- Learn to use these techniques well enough to build not just *working* systems, but *elegant* systems.

Impart knowledge, and (perhaps) develop understanding!

- Explore the design space of O-O languages, using Smalltalk and Grace fixed points

- Discuss some of the ongoing issues of Object-Oriented Programming:

- Patterns and objects?

- Whither inheritance?

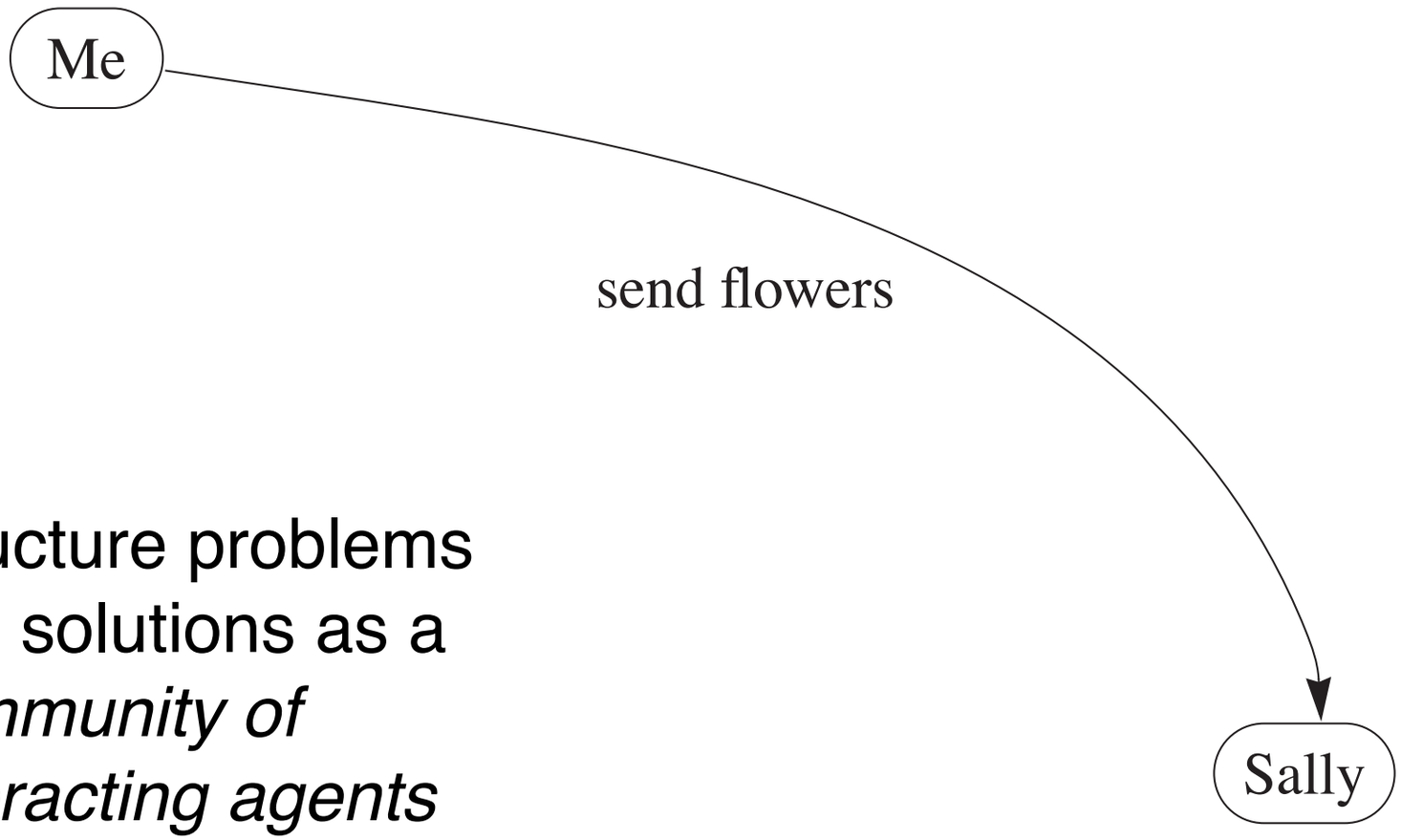
- Up-front design *vs.* Continuous design (XP)

Non-Goals

- To become an expert in Squeak (or Grace) class libraries for windowing, gui construction, or whatever
  - you will use them as tools and examples
- To become expert in OO modeling and analysis

Object-Oriented Thinking

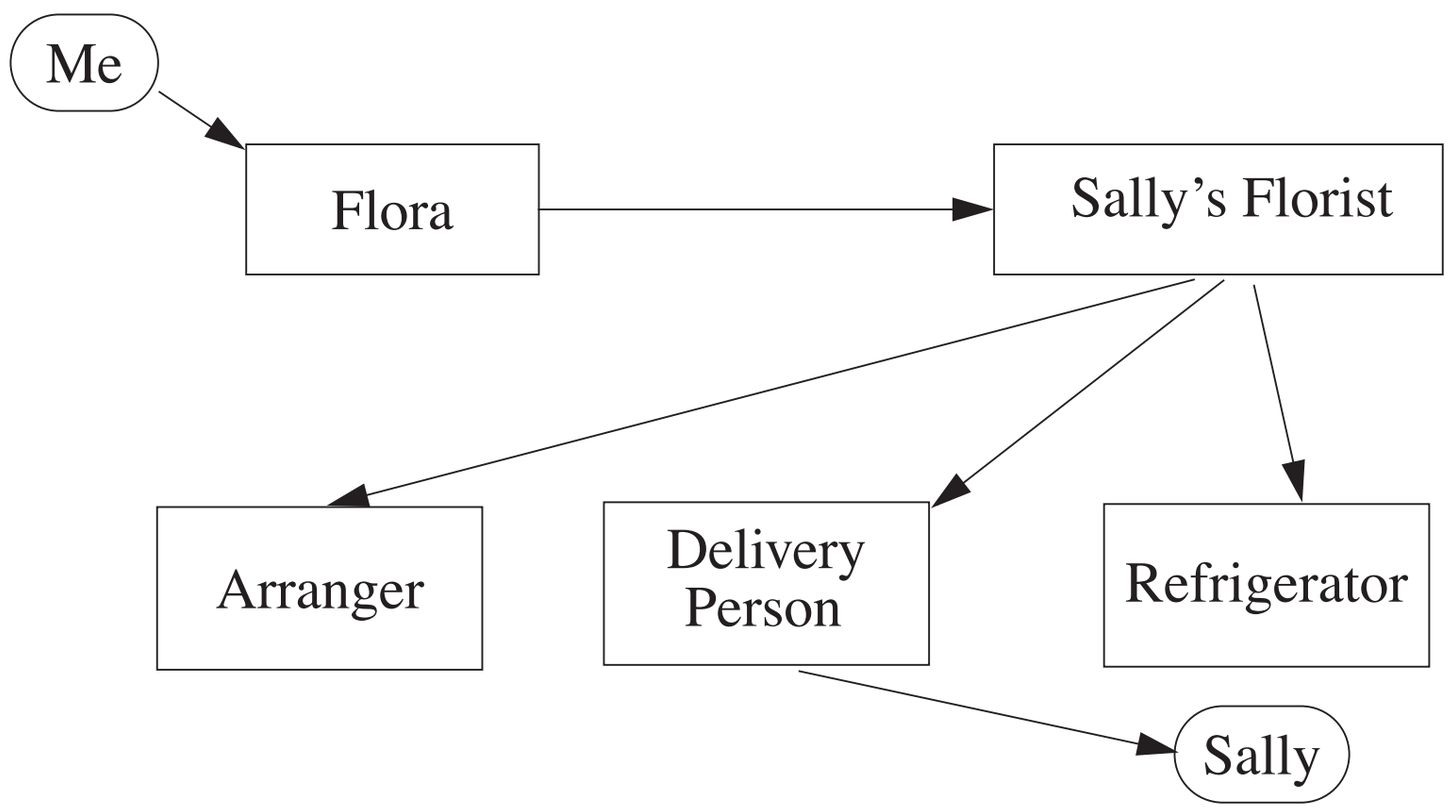
Agents & messages



- Structure problems and solutions as a *community of interacting agents*

Object-Oriented Thinking

Agents & messages



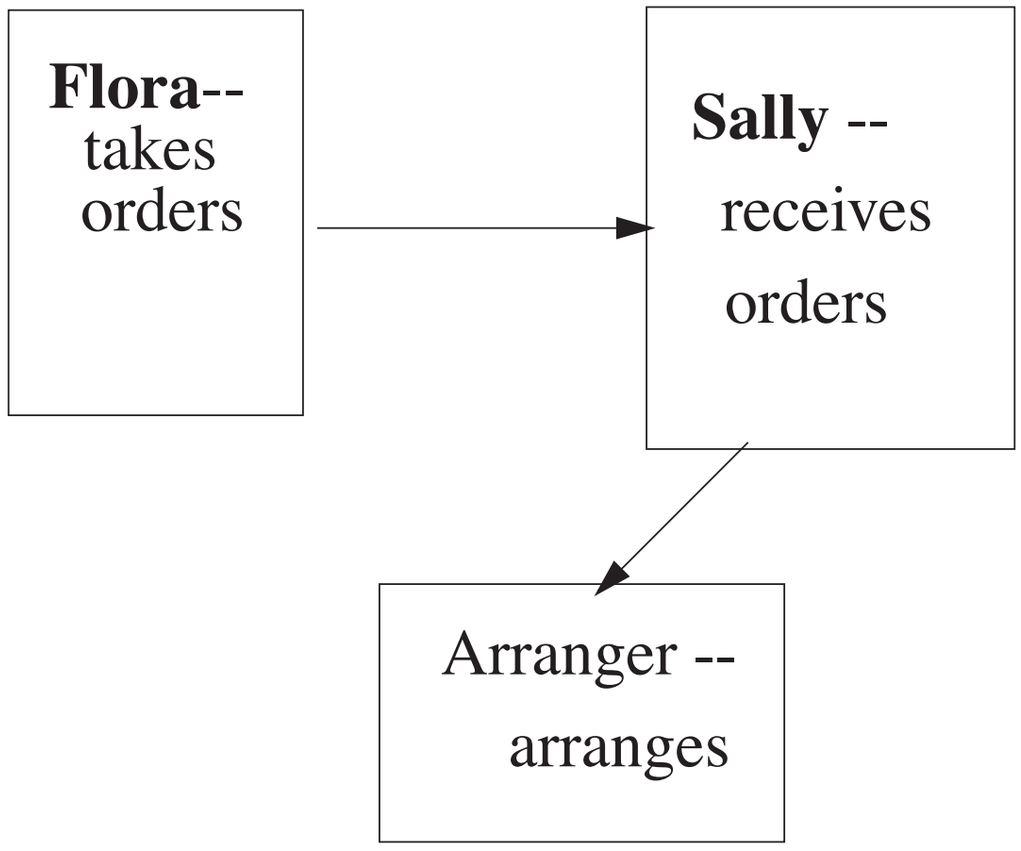
- Each agent is *autonomous*

Responsibility, Messages and Methods

An *agent* (or object) is *responsible* for achieving the desired objective.

- It delegates responsibility to other agents by sending them *messages* (requesting methods)
  - The message asks the agent to do something for me, but does not specify *how*
  - That's what "being responsible for something" *means*

Receiving agent has its own *method* to do whatever is requested

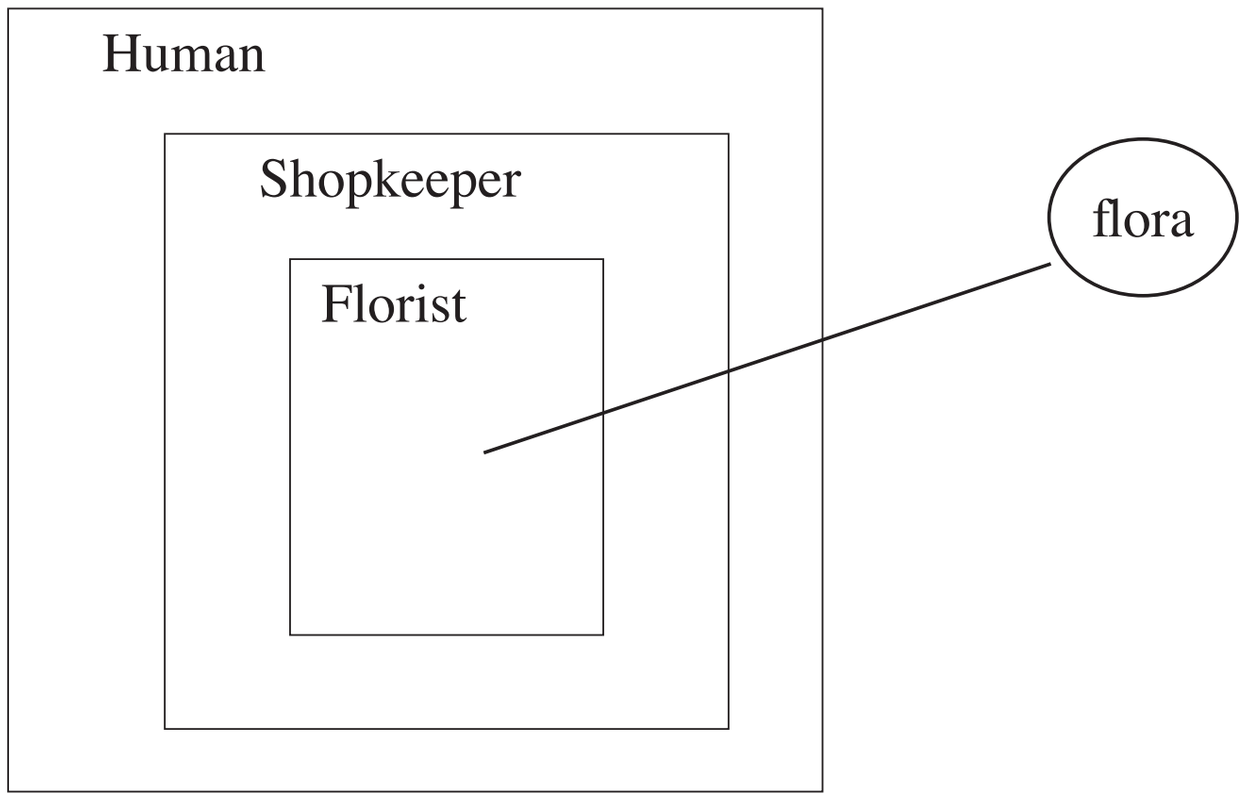


Object-oriented Thinking

Classification of Objects

- Many objects are similar to other objects
  - e.g. flora is similar to other florists
- Hence, we often define the behavior of a whole *class* of agents (Florists) rather than defining an individual (flora)
  - Not so convenient if there is only one object

– Objects often considered as members of a set of similar individuals called a *class*.



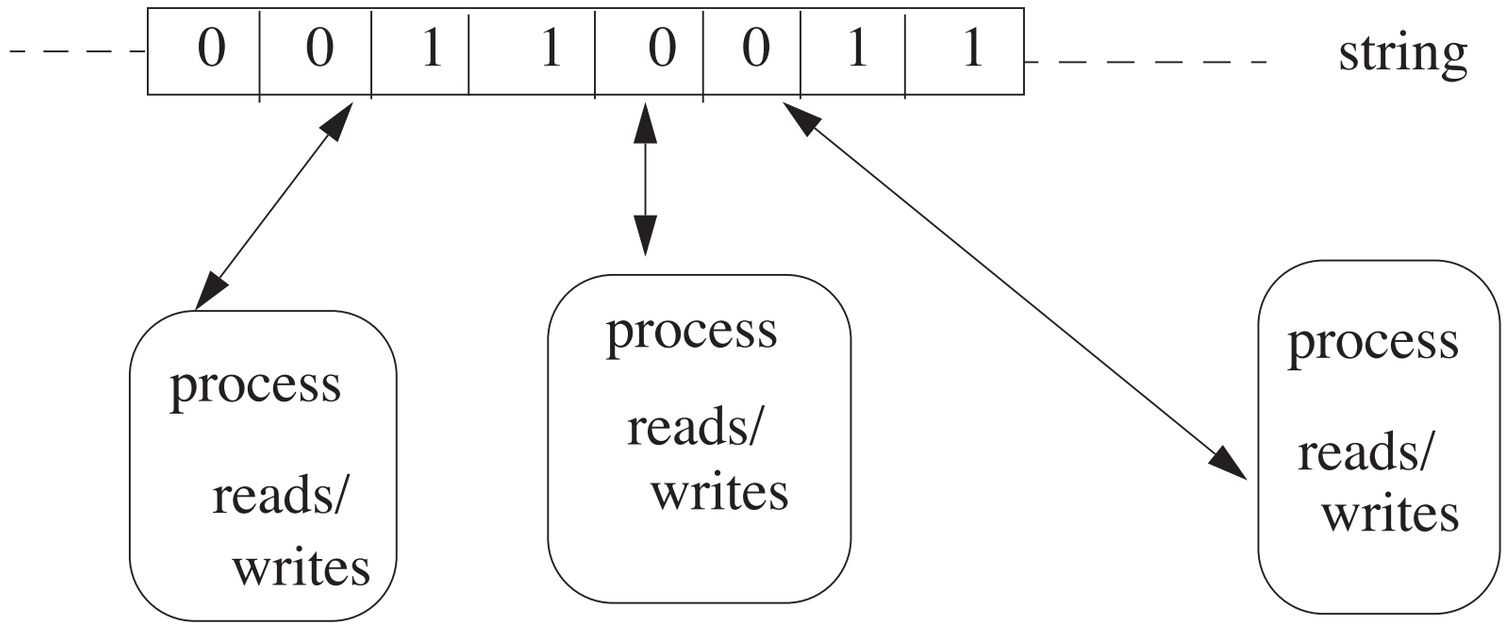
– Classes “inherit” characteristics from their parents

- some classes behave differently from their parent.

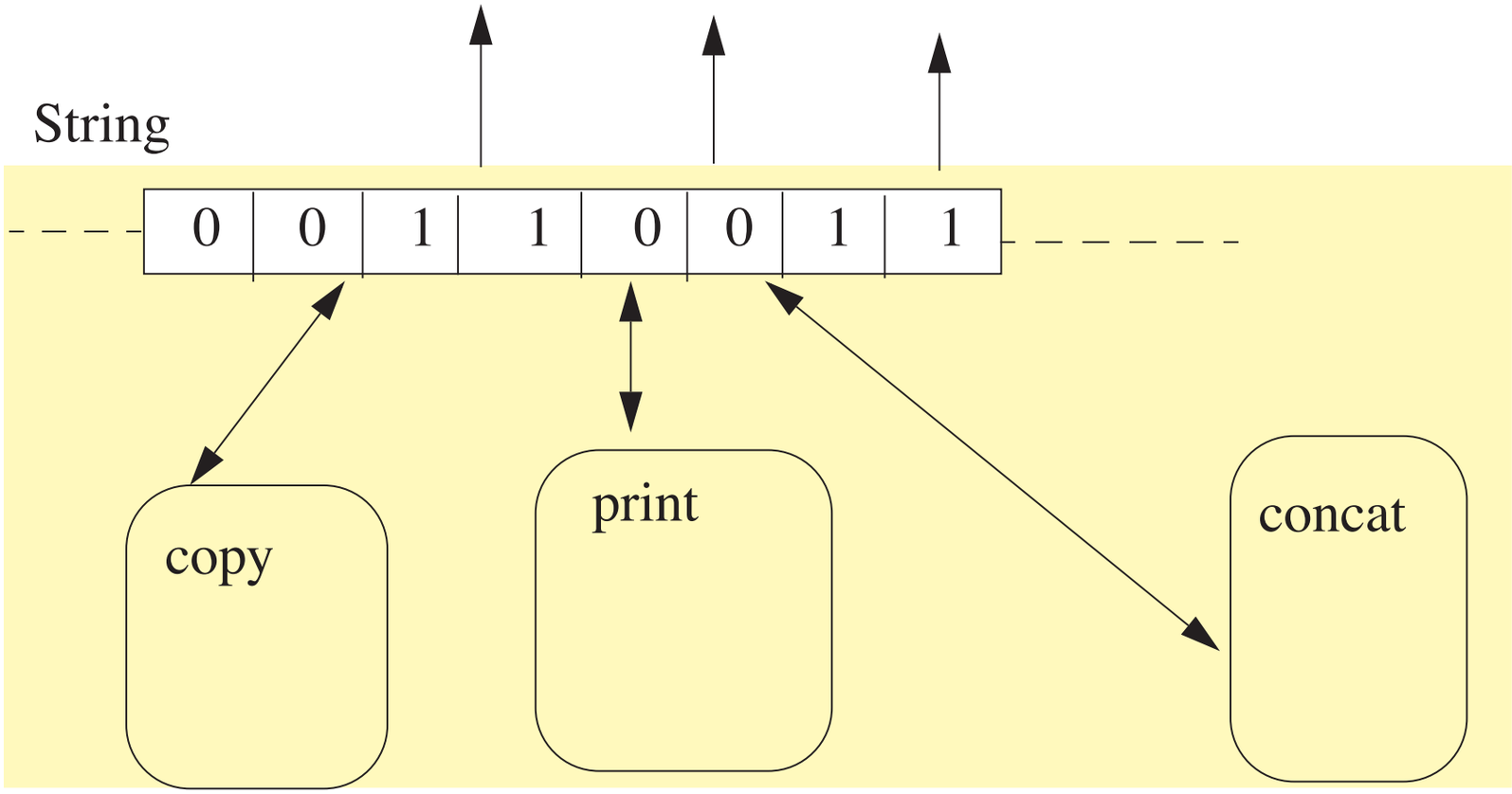
- Some objects are unique!
  - mouse, screen, uidGenerator
- Consequently:
  - some languages are class-based, some are object-based.

Objects are responsible for their own actions!

- In procedural programming, I write code that reaches into the internals of some data structure and twiddles with the bits



- In O-O programming, I politely request some other object to perform some work on my behalf, and it politely answers me



Computation as Simulation

- This collection of autonomous objects is like the world of discreet event simulation
- Anthropomorphize!
  - It's OK to think about this object talking to that object...
  - In fact, it's recommended

Programming Philosophy

- Object-Oriented programming is programming by simulation.
  - The algorithm is less important than the structure of the solution.
- When requirements change:
  - If the structure represented the structure of some ‘reality’, then the new requirements will be consistent in that reality.
  - Object-oriented design is the search for this structure: uncover the structure rather than construct in isolation.

Shopping vs. Building

- Constructing an Object-oriented application is a process of shopping for the components that one needs
  - occasionally we add a new item to the shelf.
  - usually we can find a component that *almost* fits.
- The *openness* of an OO language allows the programmer to change the component that *almost* fits into one that is a *good* fit.
  - works only if we have a rich set of components on the shelf, and if they are open to change.

Is this the *only* view of OO Programming?

No! People disagree on the meaning and role of:

1. Encapsulation
2. Types
3. Inheritance
4. Polymorphism
5. Sets and classes

Scope of CS 520/420

First 2/3 of the Quarter

- Smalltalk, programming patterns, test-driven development, continuous design

Final 1/3 of the Quarter

- Topics on languages, design patterns, library design, implementation

Ratio may vary

- Feel free to propose a topic

Pharo

Pharo is an open-source version of Smalltalk

- Derived from Squeak, in turn derived from Smalltalk-80
- Pharo workspace is a place in which you can create and interact with objects

Active community of contributors

- Runs "bit identical" on just about any platform, including many smart-phones