UML/OOA/OOD crash course

• Universal Modeling Language is a collection of standards for diagram of various aspects of problem domains and software systems.
• By itself, it doesn’t help us build software any better.
• As part of a methodical approach it is useful.
• Assignment 1 presents a design and asks you to implement it.
  – So you ought to understand the bubbles.
• I assume that many of you know UML well enough now.
  – Show of hands?
• The goal of the crash course is thus to understand one “methodical approach” the UML diagrams fit into.
Evolution of Object-Oriented Development Methods

• Mid to late 1980s
  – Object-Oriented Languages (esp. C++) were very much in vogue
  – However, there was little guidance on how to divide a problem into OO classes.

• 1990: Object Modeling
  – All at around the same time, many were borrowing an argument from structured design:
    • The best organization for a software systems is one that is cohesive in the problem domain, not in the solution space
      – Tends to isolate changes
      – Tends to make the program easier to understand
  – Developed methods for applying this concept to OO design.
    • Rumbaugh, Coad, Wirfs-Brock, Booch, Jacobson …
Object Modeling Method

• Step 1: OOA
  – Analyze the problem domain
    • Identify problem domain classes and relationships between classes
    • Identify attributes and methods
    • Identify states and transitions
    • Sample object structures and interactions
  – Not programming! Abstracting the real-world.

• Step 2: OOD
  – Use the OOA as the core of a solution to:
    • User interface design
    • Database design
    • OO program design
Time for Big Think

• If we spend the rest of this lecture on this slide, so be it.
  – Prepare yourself for some interaction! Please argue with me.
• We want to build a system whose construction in some way corresponds to the world, or problem domain, in which the system will operate.
• We want to align the concepts we use in the implementation as best we can with the concepts held by the most expert users of our system.
• We want to describe what our system should do in terms of these domain objects. (use cases)
• We want to design our software in a traceable, methodical way from what the system should do.
• OO programming is just syntax without these things.
Linkage between Domain, Design

• The transition between Analysis (Domain) and OO Design has been the stumbling block.
• Many authors have contributed and many agree at some level.
• Use cases are used to capture the sequence of operations the system must support. Related to requirements.
  – Identify many domain classes
• I like Doug Rosenberg’s ICONIX approach.
  – Robustness diagrams identify classes that support the user interface (boundary) and control logic of a system
    • Often find missed domain classes. Hence name.
    • In many cases crude notion of the UI is required to proceed.
    • http://www.iconixsw.com/uml_for_e-commerce.ppt
    • say around slide 10-17