Introduction to Object-Oriented Analysis

CMPT 371
Fall 2004
J.W. Benham
System Sequence Diagrams

• Show interactions between actor(s) and system.
  – Also possible to diagram “interesting” alternative scenarios
    • In Rational Rose, can use notes to link to basic flow
• Clarify responsibilities of system
  – Use *verbs* to show system events
• Clarify system boundary
• Created during elaboration phase
  – Possibly refined during later phases
Domain Model

• Visualize conceptual classes in problem domain, their attributes and relationships (associations)
  – Clarify vocabulary of problem domain
  – *Not* a model of software components
    • Classes in domain model may inspire software classes or other components (like subsystems)

• Also called a conceptual model, domain object model, or analysis object model
Conceptual Classes

• Symbol
  – In UML, a rectangle divided horizontally into three sections
  – Name of class in top section
• Intention – definition of conceptual class
• Extension – set of instances of that class
Identifying Conceptual Classes

• Conceptual class category list
  – See Larman, pp. 134-135

• Noun phrases in use case scenarios or other requirements documents
  – Noun phrases provide *candidates* for domain classes
  – Choose appropriate name – it will affect your view of what’s appropriate for class
What to Include in Domain Model

• Mapmaker analogy
  – Use names from vocabulary of domain
  – Exclude irrelevant features (for purpose of system under discussion)
  – Do not include classes that are not in the problem domain.
Specification or Description Classes

• Record information about a group of related items, as well as individual items
  – Information exists independently of the existence of the actual items

• Examples
  – Sales item and ProductSpecification
  – Flight and FlightDescription
Perspectives on UML Diagrams

• Conceptual: describe (possibly somewhat abstract) things in real world
• Specification: describe software abstractions independent of implementation
• Implementation perspective: describe software implementation using a particular language and technology
The Semantic Gap

• Gap between our model of problem domain and its representation in software
  – Want to minimize this gap to maximize understandability of software (even to non-programmers) so that…

• Names of classes in software should match names from domain model as closely as possible

• Software obfuscators increase this gap!