CSE 308

Objects from Use Cases

UML Tools

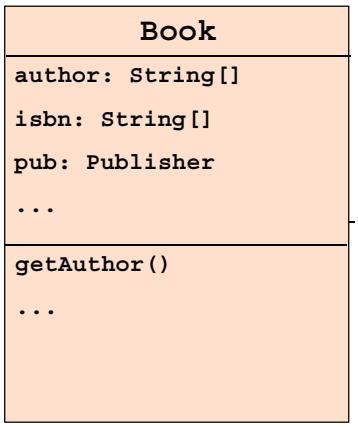
- You can use any UML tool that will generate class diagrams and sequence diagrams
  - Violet - simple, easy to use tool
    (link to download on class Web site)
  - Visio - OK, but not recommended
  - Altova Umodel - Advanced tool with 30 day free trial
    (Link in class Web site)

Object Modeling Activities

- What happens if we find the wrong abstractions?
  - Iterate and correct the model ← Do this before you write implementation code
- Steps during object modeling
  - 1. Class identification (static)
    - Based on the fundamental assumption that we can find abstractions
  - 2. Find the attributes (static)
  - 3. Find the methods (dynamic)
  - 4. Find the associations between classes (dynamic)

This essentially builds a stubbed version of your system (i.e., code structure, not implementation)

Class Notation - Reminders



Style will sometimes be determined by tool

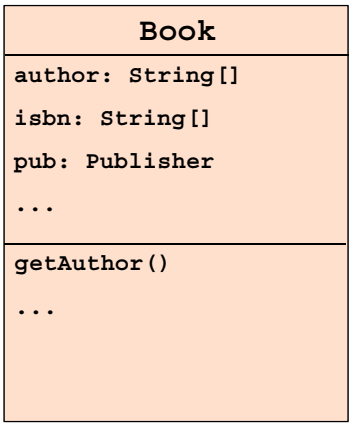
Note upper camel case for class name and lower camel case for attribute names

Class name is singular (but DB table is usually plural)

Nouns for class and attribute names and verbs for method names

Use application domain terms - not programming terms

Class Notation - Details



- Details in a class diagram will vary, based on tool, team conventions, maturity of model, etc.
- Options:
  - Parameters
  - Attribute type
  - Getter/setter methods
  - Objects in a has-a relationship
  - Method return types
  - Visibility

More details are helpful if tool generates code

Class Relationships

- Generalization / Inheritance (is-a) - dashed arrow
- Aggregation (has-a) - solid line with an empty diamond
- Composition - solid line with a filled diamond
- Multiplicity (convention may depend on tool)
  - 1..*
- Shared ownership vs. non-shared (aggregation vs. composition)

Class Identification

- The application domain has to be analyzed.
- Depending on the source (use case, GUI), different objects might be found
- Define system boundary.
  - What objects are inside, what objects are outside?
- Non-entity classes (e.g., controller, manager, and strategy) are usually difficult to immediately identify

How Do You Find Classes?

- Finding classes is the central piece in object modeling
  - Understand the application domain
  - Abbott Textual Analysis, 1983, also called noun-verb analysis
    - Nouns are good candidates for classes
    - Verbs are good candidates for operations
  - Apply design knowledge:
    - Distinguish different types of objects
    - Apply design patterns

We will cover some design patterns as they arise

Finding Objects in Use Cases

- Pick a use case and look at its scenario
  - Find terms that developers or users need to clarify in order to understand the flow of events
  - Look for recurring nouns (e.g., Incident),
  - Identify real world entities that the system needs to keep track of (e.g., FieldOfficer, Dispatcher, Resource),
  - Identify real world procedures that the system needs to keep track of (e.g., EmergencyOperationsPlan),
  - Identify data sources or sinks (e.g., Printer)
  - Identify interface artifacts (e.g., PoliceStation)
- Always use the user's terms

Object Types

- Entity Objects - tangible things
- Agents, Managers, Policies
- Events and transactions
- Users and roles
- Systems
- System interfaces and devices
- Foundational classes (String, Date, etc.)

Foundational classes are usually not included in class diagram (except possibly with inheritance)

Non-Domain Classes

- You will need to identify classes that are not associated directly with the domain (from the use cases)
- Examples
  - Controller objects (servlets) - e.g., login servlet
  - Web sharing objects - e.g., session
  - Authentication objects
  - Custom tags

Parts of Speech Mapping

Part of speech	*Model component*	*Example*
Proper noun	object	Jim Smith
Improper noun	class	toy, doll
Doing verb	method	buy, recommend
being verb	inheritance	is-a (kind-of)
having verb	aggregation	has an
modal verb	constraint	must be
adjective	attribute	3 years old
transitive verb	method	enter
intransitive verb	method (event)	depends on

Some Issues in Object Modeling

- Improving the readability of class diagrams
  - Group related classes together
  - Avoid overlapping relationship arrows
  - Break into separate class diagrams if needed
  - Eliminate non-informative attributes
- Different users of class diagrams - designers, developers
- Minimize dependency relationships
  - Minimize coupling between classes

Project Management Heuristics

- First just find objects
- Then try to differentiate them between entity, interface and control objects
- Find associations and their multiplicity
- Identify Inheritance: Look for a Taxonomy, Categorize
- Identify Aggregation
- Allow time for brainstorming

Iterate, iterate

Who Uses Class Diagrams?

- Used by:
  - The application domain expert uses class diagrams to model the application domain
  - The developer uses class diagrams during the development of a system,that is, during analysis, system design, object design and implementation

customer and the end user are often not interested in class diagrams - they focus more on the functionality of the system

Class Packages

- Group classes into discrete physical units
- Ideally use one package for each subsystem
- design principles for packaging
  - Minimize coupling
  - Maximize cohesiveness

No use of the default package in CSE308

Summary

- Modeling vs reality
- System modeling
  - Object / dynamic model
- Object modeling is the central activity
  - Class identification is a major activity of object modeling
  - There are some easy syntactic rules to find classes/objects
- Different roles during software development

Class Exercise - Volunteer Group

- Display a non-trivial use case
- Extract classes and attributes from the use case
- Use the GUI to identify additional classes and attributes