CLASS DIAGRAMS

Software Engineering

1

Reference

- Class diagrams

  `en.wikipedia.org/wiki/Class_diagram`

2

System Design Issue

- For your server, do you create a set of objects for each of your states and then construct JSONs from those objects to respond to user requests or
- Do you have your controller fetch the JSON directly from the Mongo DB to respond to user requests?

If you take the second option, how do you avoid multiple requests to the DB for the same JSONs?

3

Class Diagram

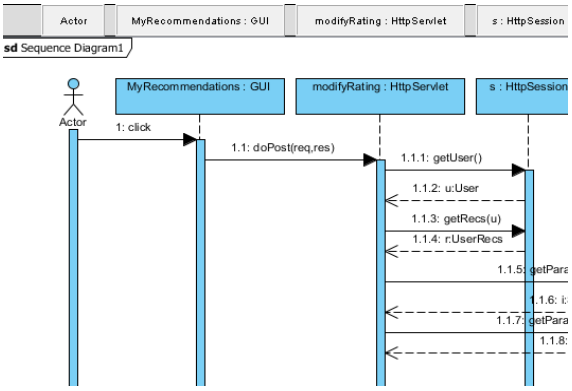
- Goal is to convey information about the static structure of your application domain
- Best if built iteratively
- Frequently, it is not a precise representation of the software structure
- Conventions you follow are largely tool and software organization based

Lucid Chart appears to be the best free tool for building class diagrams

4

UML Tools

- You can use any UML tool that will generate class diagrams and sequence diagrams
  - LucidChart – most suitable (link in "Development Tools" section of class Web site main page)
  - Visual Paradigm (14) – Community Edition has limitations
  - Violet – simple, easy to use tool (link to download on class Web site)
  - Altova Umodel – Advanced tool with 30 day free trial (Link in class Web site)



5

Object Modeling Activities

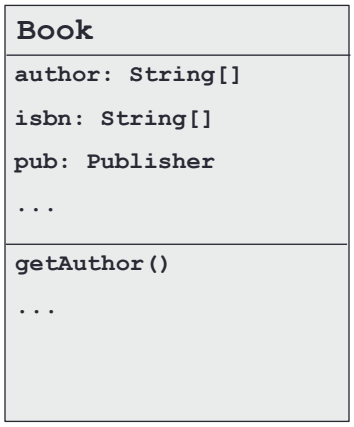
- What happens if we find the wrong abstractions?
  - Iterate and correct the model
- Steps during object modeling
  - 1. Class identification
    - Based on the fundamental assumption that we can find abstractions
  - 2. Find the attributes
  - 3. Find the methods
  - 4. Find the associations between classes

Do this before you write implementation code

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

6

Class Notation - Reminders



Style will sometimes be determined by tool

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

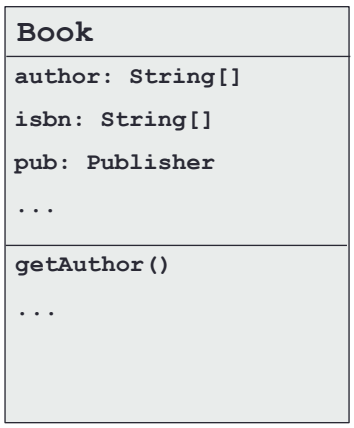
Class name is singular
(but DB table is usually plural)

Nouns for class and attribute names and verbs for method names

Use application domain terms –
not programming terms

7

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

8

Class Relationships

- Generalization / Inheritance (is-a) –arrow
- Aggregation (has-a) – solid line with an empty diamond
- Composition (owns a) – solid line with a filled diamond
- Multiplicity (convention may depend on tool)
  - 1..*

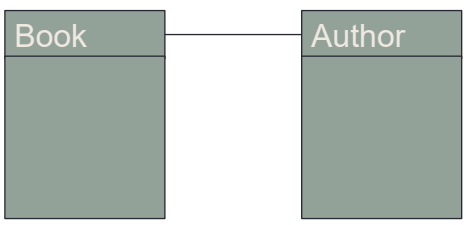
Shared ownership vs. non-shared is less important initially (aggregation vs. composition)

9

Association

- Not a statement about data flows, key relationships, etc.
- At least one class makes reference to the other
- Used when the relationship is not transient
- Options
  - Named
  - Multiplicity
  - Diamond (showing ownership)
  - Other properties

A Book has an Author



10

Association Arrowhead

- Usually means that the class at the tail of the arrow has an attribute of the type (Class name) shown at the end of the diamond arrow
- Domain UML associations do not use arrowheads (SW UML does)

11

How to Express Attributes

- Choices
  - Attribute text
  - Association lines
- Guidelines
  - Attribute text for primitive types
  - Attribute text for library class types
  - Association lines for class types

Not considered incorrect to show both attribute text and an association line

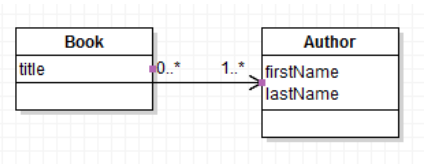
12

Multiplicity

- Multiplicity in an association indicates the number of instances
- Multiplicity symbol is often tool-related

UML tools allow you to add labels to an association

Symbol	Instances
0..1	No instances or one instance
1	Exactly one instance
0..*	Zero or more instances
1..*	One or more instances
3,5,8	Exactly 3, 5, or 8



Note, you might not include related classes in attributes

13

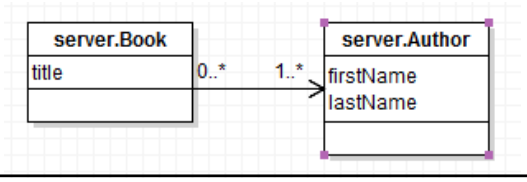
Methods

- Typically, developed later in the design phase
- Sequence diagrams are very helpful in determining needed methods
- No need to include obvious methods (e.g., getters and setters)
- Class diagram might include parameters and return type

14

Package

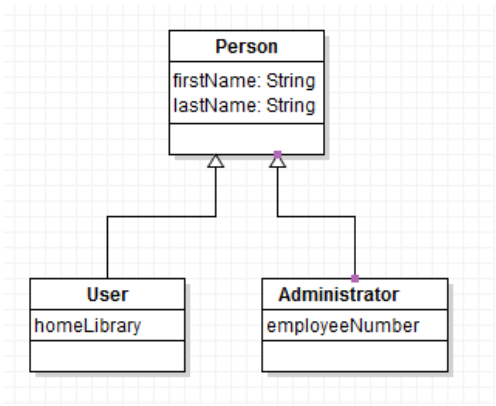
- If you are showing multiple packages in a single class diagram, either
  - Surround the package classes with a dashed border
  - Include your package identifier in the Class name
- Be sure that your packages are organized logically to maximize cohesion
- Do not use the default package



15

Inheritance

- More general term is Generalization



16

Keywords

- Textual adornment to categorize a model element
- Can be shown in double brackets (<<…>>) or curly braces ({…})
- Examples
  - Interface
  - Abstract

17

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

18

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

19

Finding Objects in Use Cases

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

20

Object Categories

- 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)

21

Non-Domain Classes

- You will need to identify classes that are not associated directly with the domain (from the use cases)
- Examples
  - Controller objects – e.g., request handler
  - Web sharing objects – e.g., session
  - Authentication objects
  - Resource managers

22

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 and methods (e.g., getter methods)
- Different users of class diagrams – designers, developers
- Minimize dependency relationships
  - Minimize coupling between classes

23

Project Management Heuristics

- First just find objects
- Find associations and their multiplicity
- Identify Inheritance: Look for a Taxonomy, Categorize
- Identify Aggregation
- Allow time for brainstorming
- Be flexible in changing your design, if needed

Iterate, iterate, iterate

24

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

25

Class Packages

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

Use of the default package in your server design in CSE416 is not permitted

26

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

27