

The University of Iowa

**CS:2820 (22C:22)**

**Object-Oriented Software  
Development**

Spring 2015

**Object Oriented Analysis  
and Design**

# Educational Goals

- Apply principles and patterns to create better object-oriented software designs
- Iteratively follow a set of common activities in analysis and design
- Use an agile approach to the Unified Process as an example
- Create frequently used diagrams in the UML notation

# What is Analysis

- An investigation of the problem and requirements (not of solutions)
  - requirements analysis
  - object-oriented analysis
- Goal: *do the right thing*

# What is Design

- development of a conceptual solution (in software and hardware) that fulfills the requirements
  - object-oriented design
  - database design
- No low-level details
- Goal: *do the thing right*

# OO Analysis and Design

## **Object-oriented analysis:**

emphasis on finding and describing the objects, or concepts, in the problem domain

## **Object-oriented design:**

emphasis on defining software objects and how they collaborate to fulfill the requirements

# Unified Modeling Language

- a visual language for

- specifying,
- constructing, and
- documenting

the artifacts of a system

- *de facto* standard for object-oriented software development

# Uses of UML

- As a sketch
- As a blueprint
- As a programming language

# UML as a Sketch

Informal and incomplete diagrams  
created to explore difficult parts of the  
problem or solution space

# UML as a Blueprint

Relatively detailed design diagrams used either for

1. *reverse engineering*, to visualize and better understand existing code, or
2. *forward engineering*, to drive code generation

# UML as a Programming Language

Complete executable specification of a software system in UML

- Executable code automatically generated
- Code not normally seen or modified by developers
- technology not quite mature yet

# UML Perspectives

1. Conceptual perspective
2. (Software) Specification perspective
3. (Software) Implementation perspective

# Conceptual Perspective

- UML diagrams describe entities in the real world or domain of interest

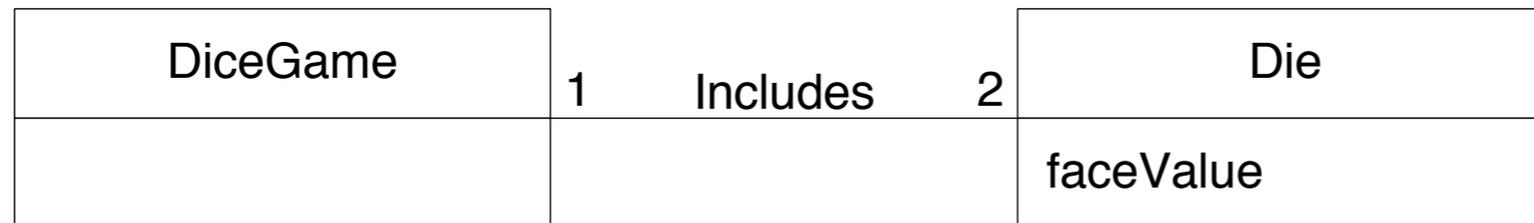
# Specification Perspective

- UML diagrams describe software abstractions or components with specifications and interfaces
- There is no commitment to a particular implementation  
E.g., not specifically a class in Scala or Java

# Implementation Perspective

- UML diagrams describe software implementations in a particular technology

e.g., Java



**Conceptual Perspective  
(domain model)**

Raw UML class diagram notation used to visualize real-world concepts.



**Specification or  
Implementation  
Perspective  
(design class diagram)**

Raw UML class diagram notation used to visualize software elements.

# Credits

Notes and figures adapted from

*Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* by C. Larman. 3rd edition.  
Prentice Hall/Pearson, 2005.