On behalf of ASE crew, I welcome you onboard. We will cover this journey in roughly 50 minutes.

We will fly over the basics of Object Oriented Analysis and Design. We may encounter some turbulence at times. Since you don't have seatbelts, please stay awake.

Thank you for choosing to attend this tutorial and wish you a pleasant flight (lecture)!

--Venkatesh Vinayakarao

Object Oriented Analysis and Design (OOAD)

An Introduction

Venkatesh Vinayakarao
venkatesh.v@iiits.in
http://vvtesh.co.in

Assistant Professor
Indian Institute of Information Technology, Sri City, Chittoor

The art of simplicity is a puzzle of complexity. **–Douglas Horton.**

House Rule

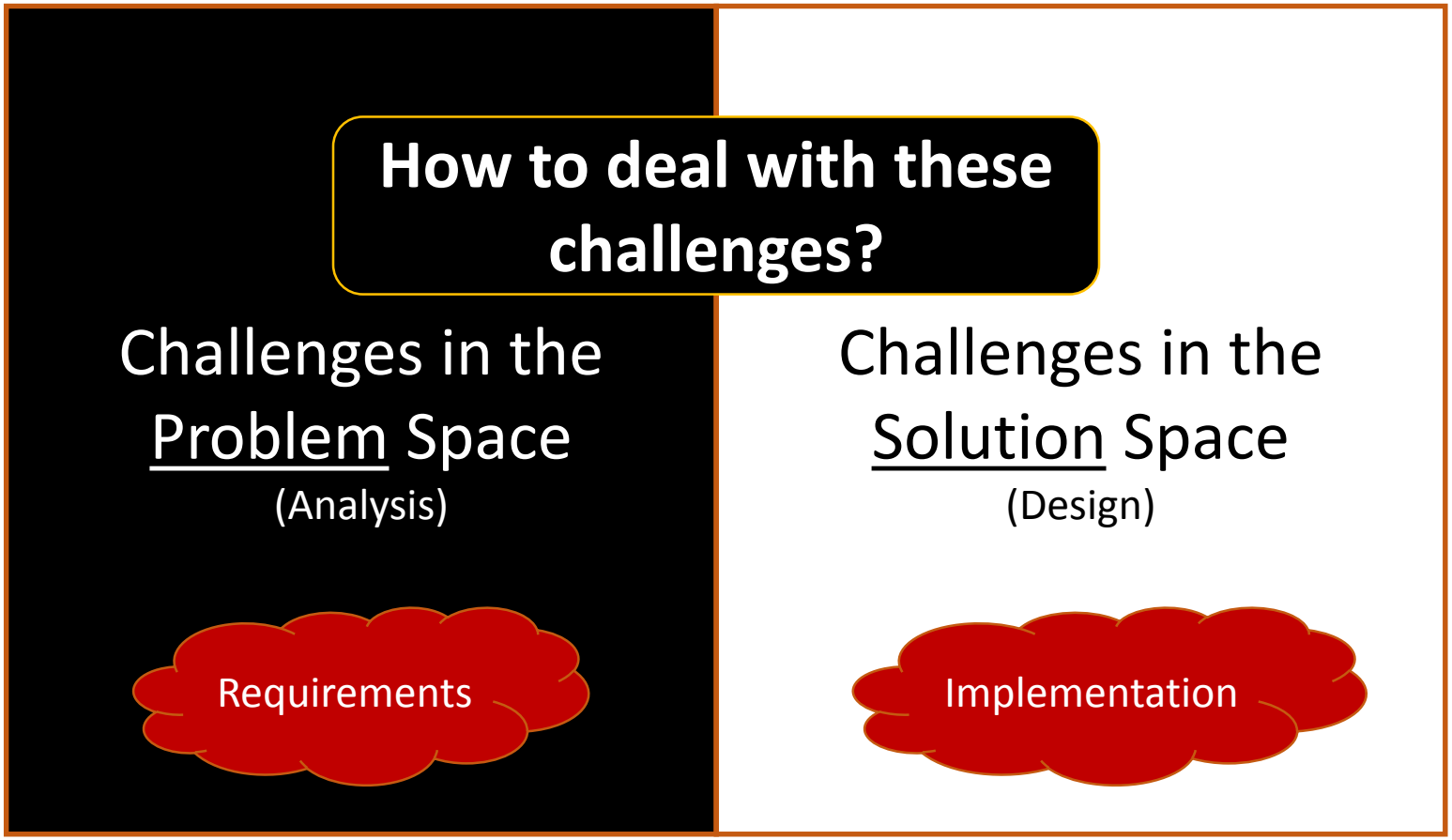
- Raise your hand if you know the answer.
- Do not shout out unless offered a chance by your instructor.

Software Engineering (SE) is very challenging! Why?

Challenges in SE

- Over-specification and under-specification
- Relying on star performers
- Weak personnel and problem employees
- Requirements creep
- Code not even understood by the author – done by trial and error
- Designing, Coding, …
- …
- …. and >100 more!

Challenges in RE



How to tame complexity?

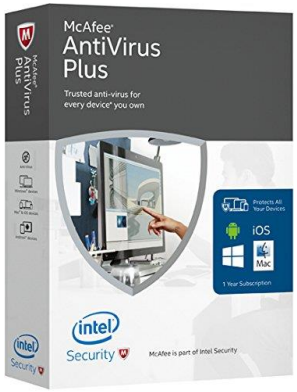
Can we learn from the world around us?

How to get All India Rank 1 in JEE?

*Am not AIR 1 in JEE. Just for fun.

How did we humans build this?

We could build these too!



How to tame complexity?

Can we learn from the world around us?

Taming Complexity

Key Principles

1. Hierarchy
2. Abstraction
3. Keeping Related Things Together
4. Polymorphism

Do you recognize this?

*Table of Contents of C Programming Book

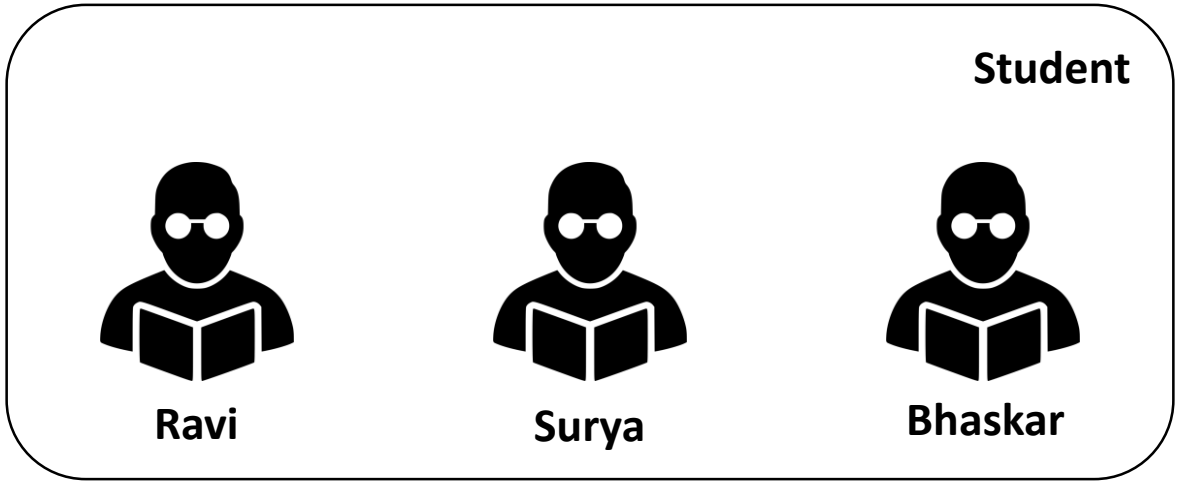
Objects are Everywhere!

- We are inherently Object Oriented!
- An Example Object:
  - Identity (Venkatesh)
  - Data (Height = 5’ 6’’, Weight = 80 KG*, …)
  - Behavior (Lecture, Put Students to Sleep, …)

Objects have Identity, Data and Behavior associated with them.

*May be slightly more. ☺

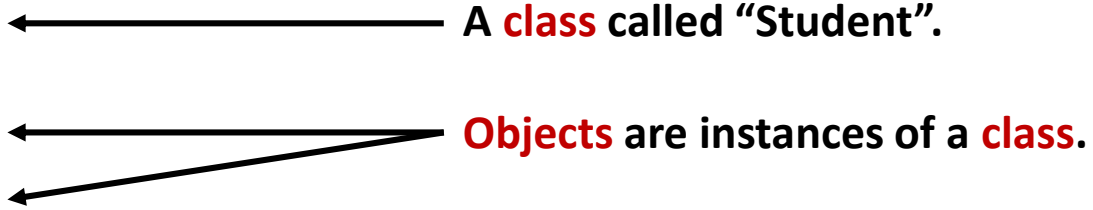
Abstraction



"*Student*" is a class.
The objects namely "*Ravi*", "*Surya*" and "*Bhaskar*" belong to the "Student" class.

Abstraction

- Classroom
  - Classroom 101
- Student
  - Student1
  - Student2
  - …
- Faculty
  - Faculty1
- Electrical Fitting
  - Light
    - Power Saver Light
      - PSL 1
      - PSL 2 … and so on.



Class Vs. Object

- Class has:
  - Data (Dogs are four legged [quadruped]).
  - Behavior (Dogs bark).
- Object has:
  - Identity (Jimmy).
  - Data (four legged).
  - Behavior (barks).

Quiz

- Which of the following is not a "**class**"?
  - Employee
  - Train
  - Vehicle
  - India

Quiz

- Which of the following is not a "**class**"?
  - Employee
  - Train
  - Vehicle
  - **India**

Quiz

- Which of the following is not an "Object"?
  - LG Q6 Mobile Phone
  - Subu Kandaswamy
  - Pinakini Express
  - Employee with ID 231

Quiz

- Which of the following is not an "Object"?
  - **LG Q6 Mobile Phone**
  - Subu Kandaswamy
  - Pinakini Express
  - Employee with ID 231

Three Volunteers Please…

Describe this!

Why UML?

Part of the problem in industry today is related to

1. Specification
2. Visualization
3. Construction

What is UML?

Unified Modeling language (UML)

is a

standardized *modeling language*

enabling developers to

specify, visualize, construct and document

artifacts of a software system.

History

•UML combines best of three principal methods:

- The Booch method, devised by Grady Booch,
- Object-oriented Modeling Technique (OMT), devised by Jim Rumbaugh,
- Object-oriented Software Engineering (also known as Objectory), devised by Ivar Jacobson.

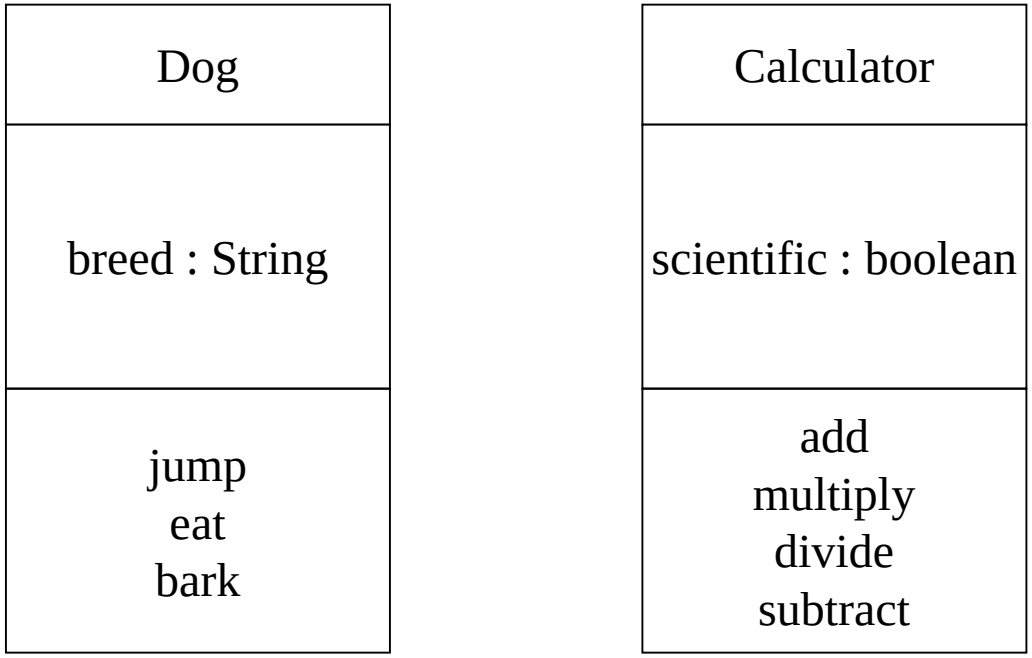
History

- **1994**
  - Jim Rumbaugh joins Grady Booch at Rational Software to merge their methods.
- **1995**
  - Booch and Rumbaugh published version 8 of the Unified method. Rational Software buys Objectory and Ivar Jacobson joins the company.
- **1997**
  - Booch, Rumbaugh and Jacobson release (through Rational) a proposal of version 1 of UML.
- **1997**
  - UML version 1.1 was adopted by The Object Management Group (OMG), a non-profit organization.

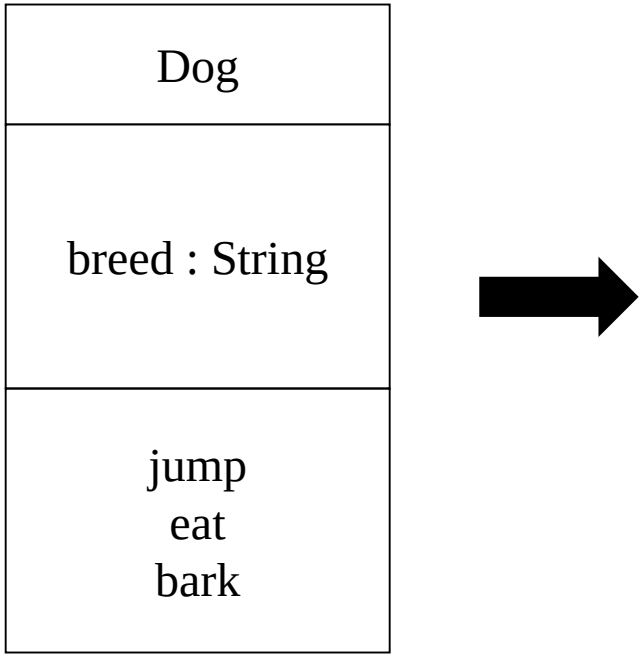
Modeling Software Systems

- How is the software structured? (Structural Description)
  - **Class Diagram**
  - **Object Diagram**
  - Component Diagram
  - Deployment Diagram
  - Composite Structure Diagram
  - Package Diagram
- What does the software do? (Behavioral Description)
  - Use Case Diagram
  - Activity Diagram
  - Interaction Overview
  - How do multiple components interact? (Interaction Description)
    - *Sequence Diagram*
    - *Communication Diagram*
    - *Timing Diagram*
    - *Interaction Overview Diagram*

Class Notation



Class → Code Transformation



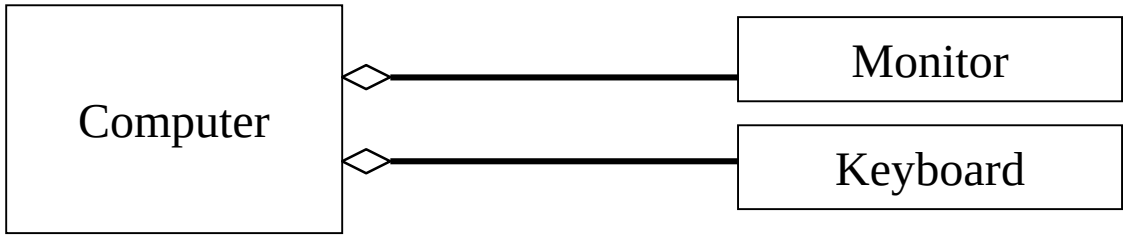

```
public class Dog
{
    private String breed;

    public int bark()
    {
        ...
    }

}
```

Classes have Relationships

Association

Relationships - Aggregation

- *Aggregation* represents a part-whole or part-of relationship.



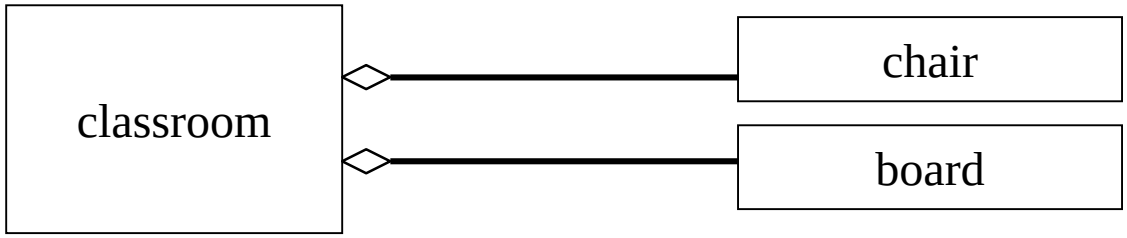
Examples are taken from OMG tutorial on UML by Cris kobryn

Quiz

- Can you identify an aggregation relationship for this classroom?

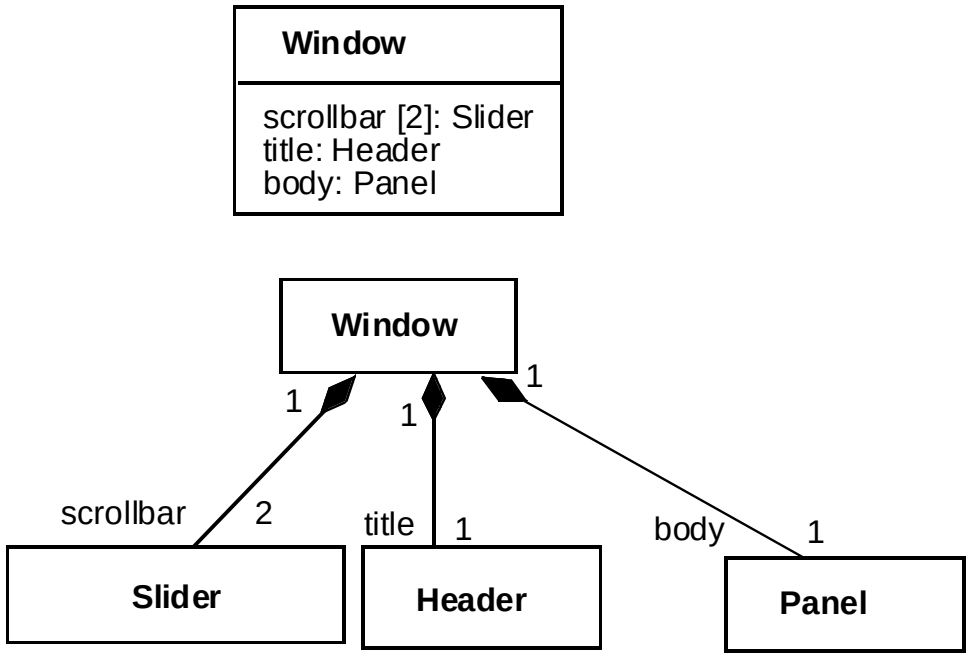
Quiz

- Can you identify an aggregation relationship for this classroom?



Relationships - Composition

- "owns a" or association relationship
- Part dies with the whole.
- Note that, window may exist without a slider!

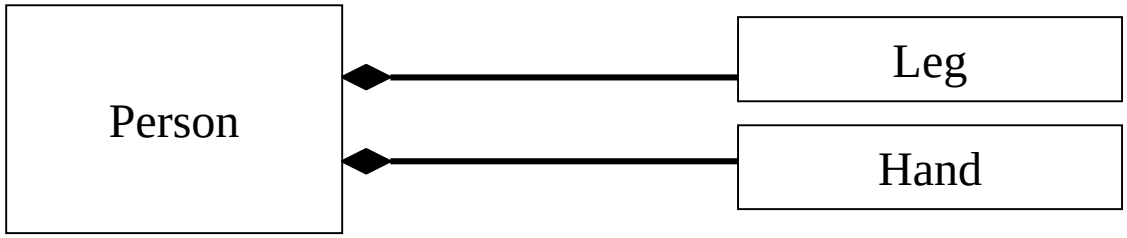


Quiz

- Can you identify any composition relationship?

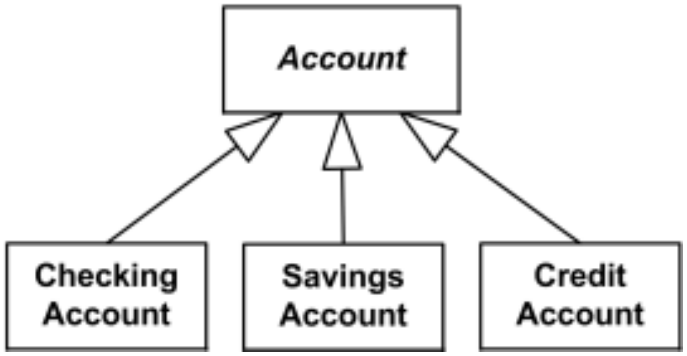
Quiz

- Can you identify a composition relationship?

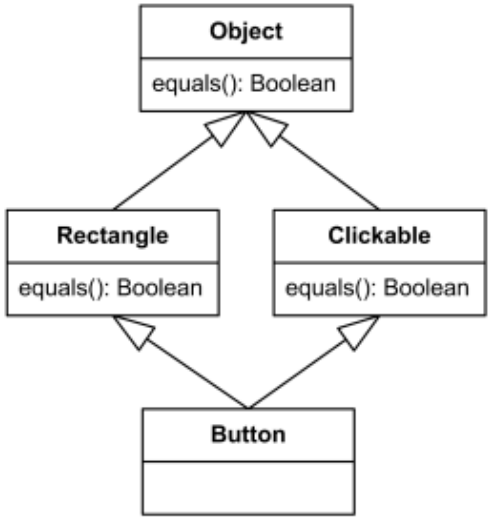


Relationships - Generalization

- **Supertype – subtype** relationship.
- Also known as "**is a**" relationship.
- In practice, this means that any instance of the subtype is also an instance of the



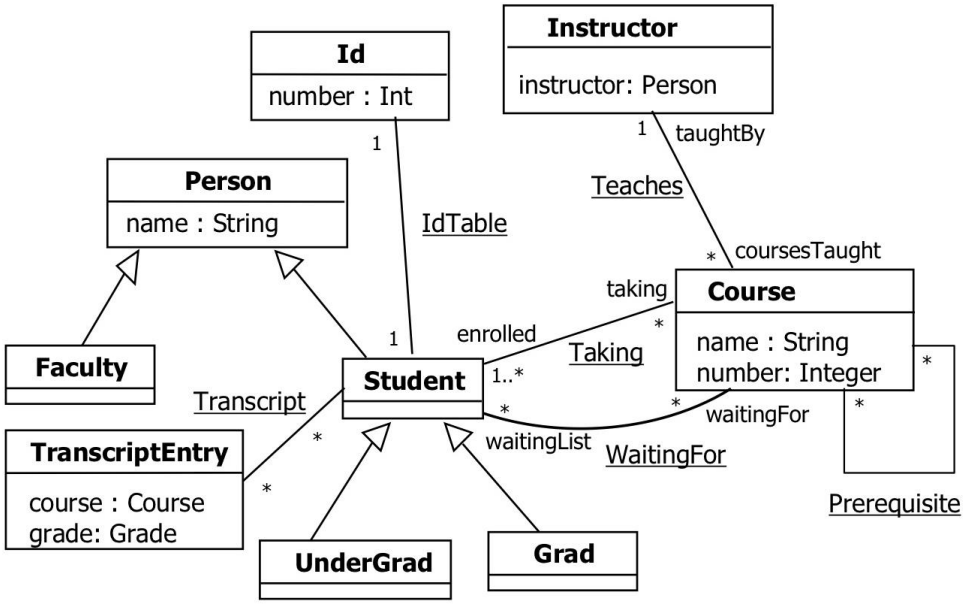
Example 2: There are three account types: Checking, Savings and Credit.



Example 1: Button is a rectangular clickable object.

Class Diagram

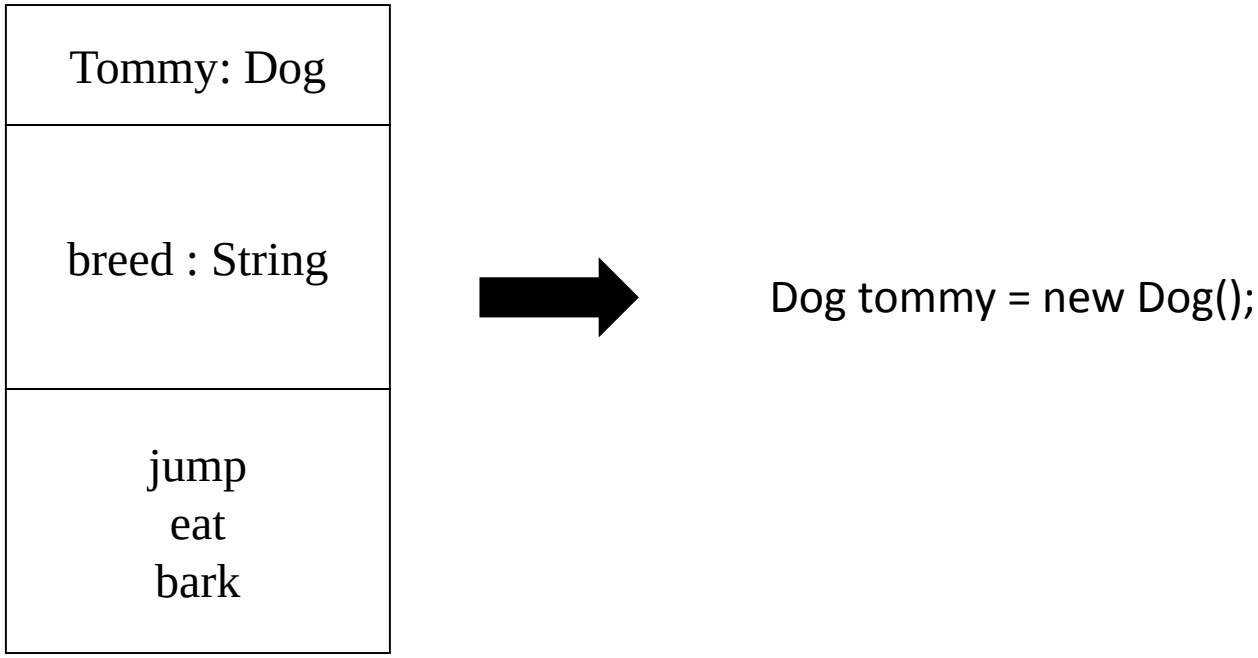
University Scenario



Source: uiowa.edu.

Object Diagram

- At a specific time, shows the object instances and relationships.

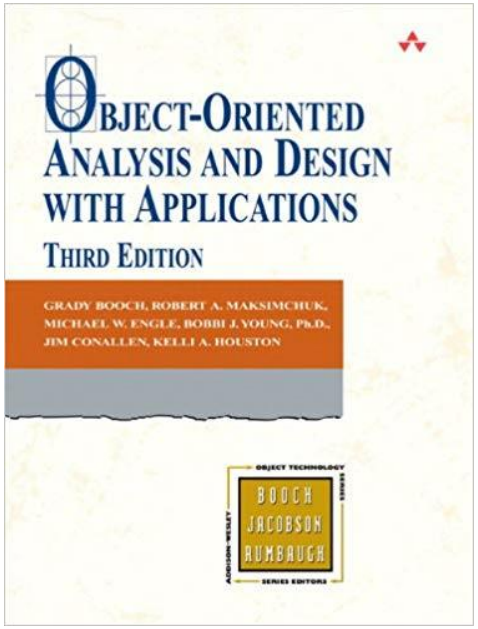


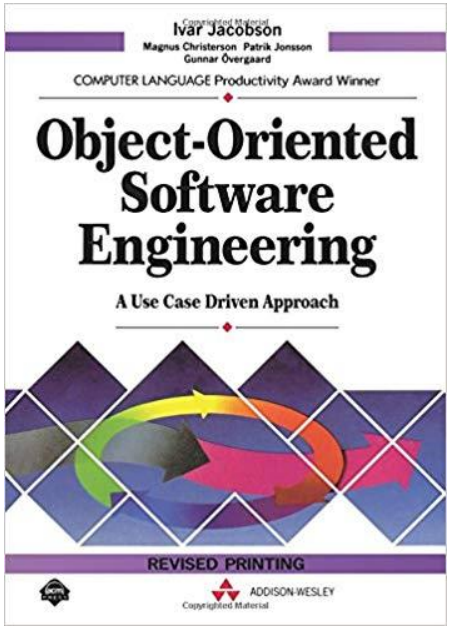
Limitations

- Complex to hand-write UML diagrams. We need tools.
- Even with UML, auto-generation of code requires the model to be at very low level. This is considered impractical.
- There are too many diagrams and yet descriptions are not well captured.

Resources

- Tools: ArgoUML (http://argouml.tigris.org/)
- Books:



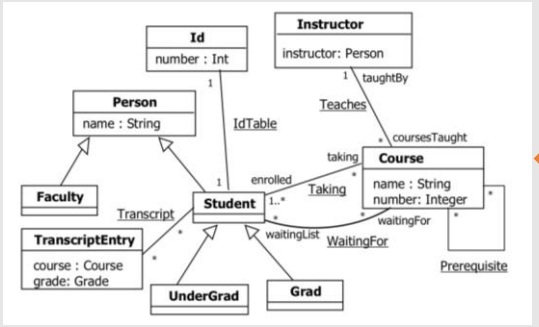


Key Principles

1. Hierarchy
2. Abstraction
3. Keeping Related Things Together
4. Polymorphism

Class Vs. Object

- Class has:
  - Data (Dogs are four legged [quadruped]).
  - Behavior (Dogs bark).
- Object has:
  - Identity (Jimmy).
  - Data (four legged).
  - Behavior (barks).



Unified Modeling language (UML)
is a
standardized *modeling language*
enabling developers to
specify, visualize, construct and document
artifacts of a software system.

THANK YOU

The art of simplicity is a puzzle of complexity. **–Douglas Horton.**