Introduction

Idaho State University | Computer Science

Dr. Isaac Griffith

CS 2263
Department of Computer Science
Idaho State University

ROAR

Outcomes

After today's lecture you will:

- Have an understanding of the course syllabus
- Have an understanding of the course layout
- Have an understanding of the importance of OOAD
- Have an abridged understanding of the history of OO

Course Introductions

CS 2263

Introductions

- Let's go around the class
  - Your name
  - Your major
  - Your thoughts on what this class is about.

Syllabus Review

CS 2263

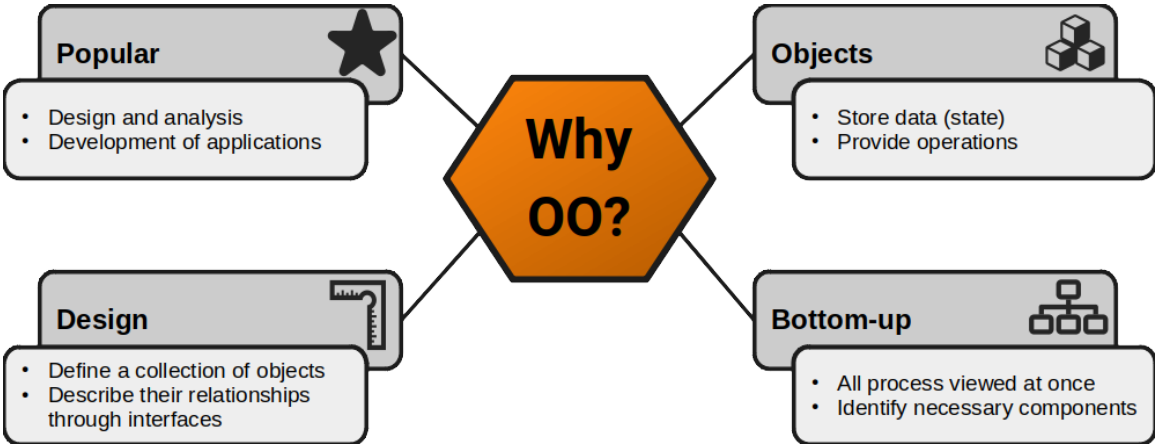
ROAR

Introduction to OOAD

CS 2263

ROAR

Why Object-Oriented



Two Perspectives

Procedural	Object-Oriented
Focuses on processes, algorithms, and data structures	Focuses on behavior and relationships
Design uses a top-down decomposition into functions	Design uses a holistic bottom-up composition into objects approach
Changes in data or function often lead to changing the program	Changes in data or function are easier to accommodate
No method for hiding data and thus less secure	Provides data hiding and is more secure
Function favored over data	Data favored over function

Key Ideas of OO

Central role of Objects

- Objects are first-class
- Data and operations
- Processes rely on Objects

Notion of a class

- Classes **type** Objects
- Categorize, classify, and define hierarchies

Abstract Spec of Function

- **Interface -** Abstract specs that do not restrict function
- **Abstract classes**

Standard Solutions

- **Design Patterns**
- All phases of SDLC

A Language to define the System

- **UML**
- Similar to Blueprints

Analysis process to model a system

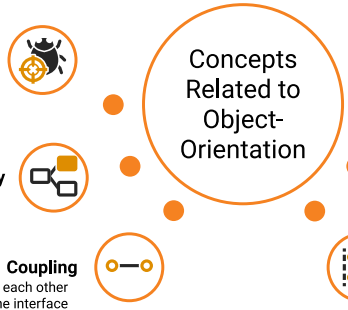
- Specs → **Conceptual Design**
- Conceptual Classes → **Implementation Classes**

Notions of Extendability and Adaptability

- Extendability → **Inheritance**
- Adaptability → **Composition**

ROAR

Related Concepts



Testability

- Degree to which a system may be easily tested.
- **High Testability** → bugs are easier to find, system facilitates bug detection

Modifiability

- Degree to which a system may be easily modified.
  - Modifications can be to function or design
  - **High Modifiabiilty** → more *adaptable* system

Coupling

- How dependent modules are on each other
- **Low Coupling** → results from relying on the interface rather than implementation.
- **High Coupling** → changes in one module necessitate changes in other modules. Reduces understandability

Modularity

- Putting together a system from independent smaller and simpler components

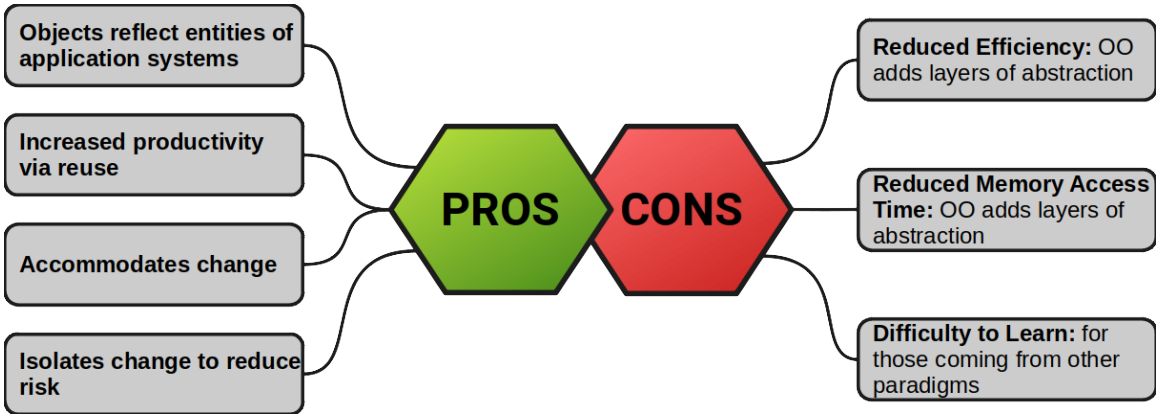
Encapsulation

- Modules should clearly specify its function without exposing its implementation to external agents.
- Example → **ADTs**

Cohesion

- How well module entities work together to provide module functionality
- How focused the module's responsibilities are
- **High Cohesion**: Improves reliability, reusability and understandability

Benefits and Drawbacks of OO

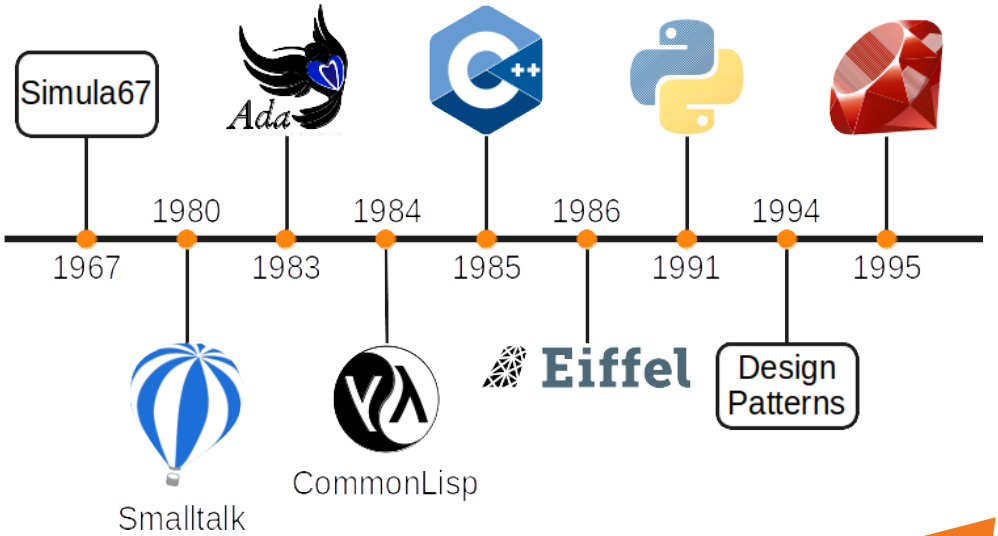


OO History

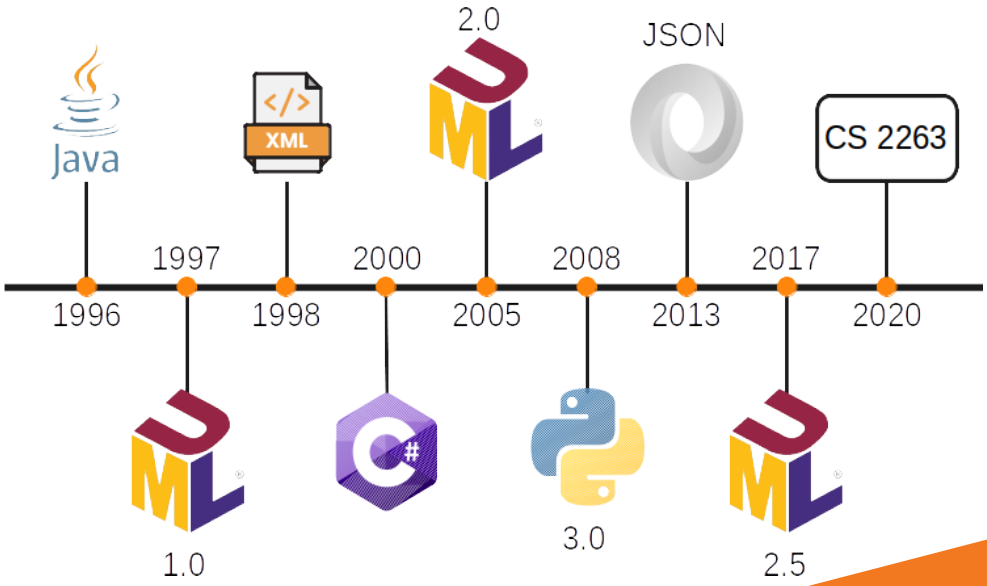
CS 2263

ROAR

History

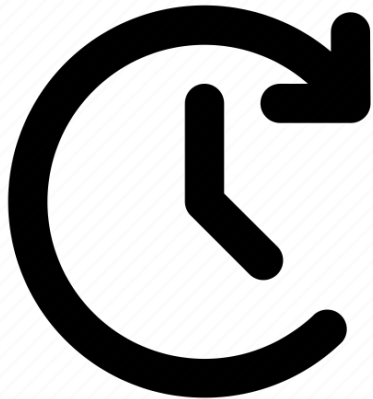

Simula67
Ada
1967
1980
1983
1984
1985
1986
1991
1994
1995
Smalltalk
CommonLisp
Eiffel
Design Patterns

History



For Next Time

- Review Chapter 1
- Review Course Syllabus
- Review the Lecture
- Read the Appendix and Chapter 2.1 - 2.2
- Make sure you have the following software installed:
  - A JDK (Oracle's or OpenJDK)
  - A professional grade Java IDE (such as IntelliJ IDEA)
- Come to Class
- Start Homework 01



Are there any questions?