speed quality

Rational®

the software development company

Mastering Object-Oriented Analysis and Design with UML

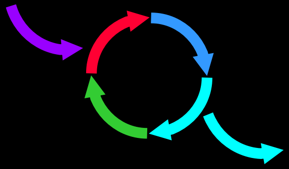
Module 2: Concepts of Object Orientation

Objectives: Concepts of Object Orientation

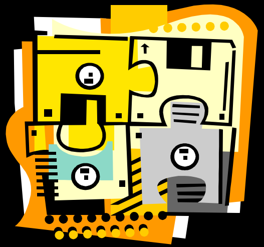
- Explain the basic principles of object orientation
- Define the basic concepts and terms of object orientation and the associated UML notation
- Demonstrate the strengths of object orientation
- Present some basic UML modeling notation

Best Practices Implementation

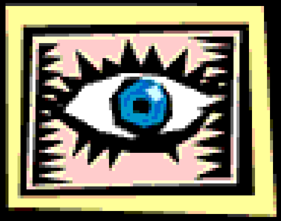
- Object technology helps implement these Best Practices.
  - Develop Iteratively: tolerates changing requirements, integrates elements progressively, facilitates reuse.



  - Use Component-Based Architectures: architectural emphasis, component-based development.

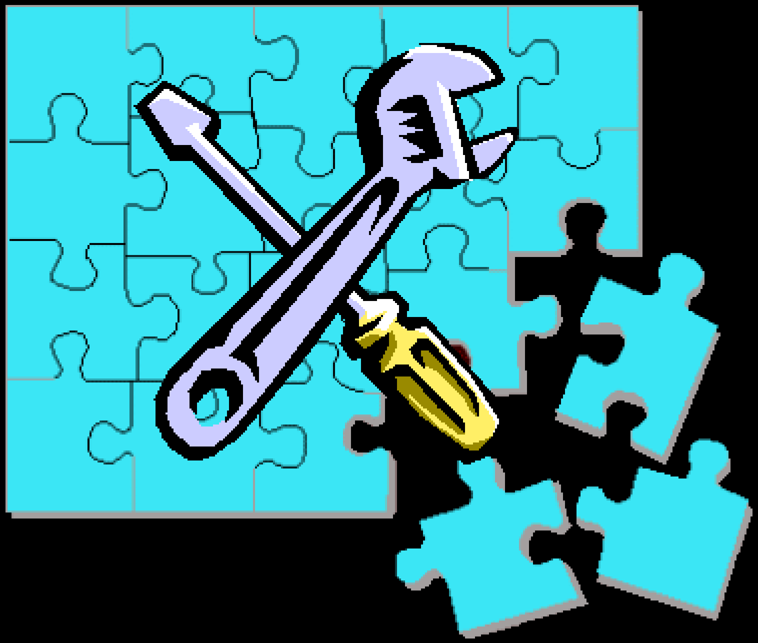


  - Model Visually: easy understanding, easy modification.



What Is Object Technology?

- Object technology
  - A set of principles guiding software construction together with languages, databases, and other tools that support those principles. (*Object Technology: A Manager's Guide*, Taylor, 1997)

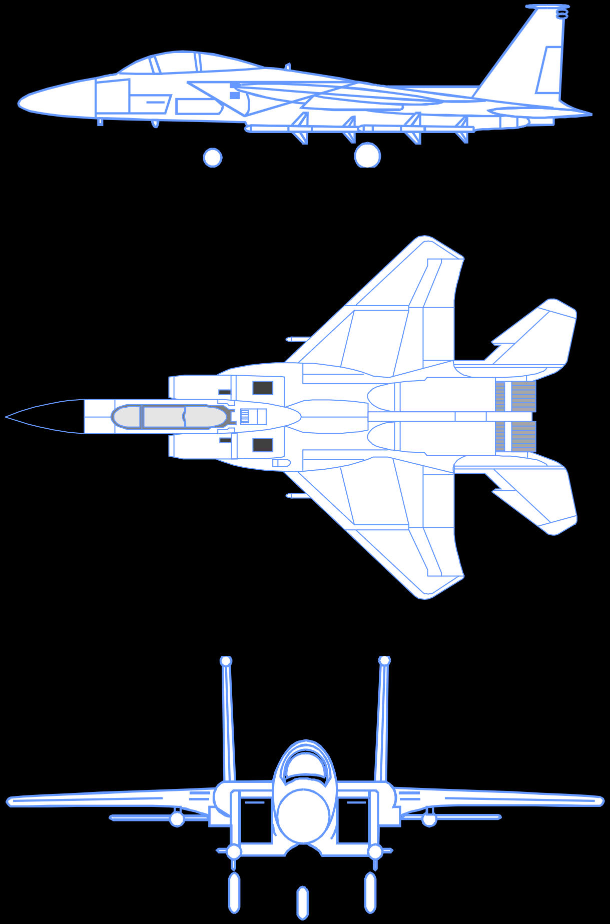


Strengths of Object Technology

- Provides a single paradigm
  - A single language used by users, analysts, designers, and implementers
- Facilitates architectural and code reuse
- Models more closely reflect the real world
  - More accurately describes corporate entities
  - Decomposed based on natural partitioning
  - Easier to understand and maintain
- Provides stability
  - A small change in requirements does not mean massive changes in the system under development
- Is adaptive to change

What Is a Model?

- A model is a simplification of reality.

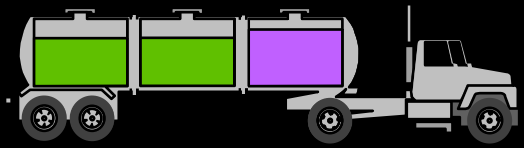


Why Do We Model?

- We build models to better understand the system we are developing.
- Modeling achieves four aims. It:
  - Helps us to visualize a system as we want it to be.
  - Permits us to specify the structure or behavior of a system.
  - Gives us a template that guides us in constructing a system.
  - Documents the decisions we have made.
- We build models of complex systems because we cannot comprehend such a system in its entirety.

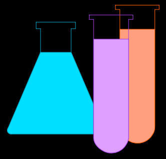
What Is an Object?

- Informally, an object represents an entity, either physical, conceptual, or software.
  - Physical entity



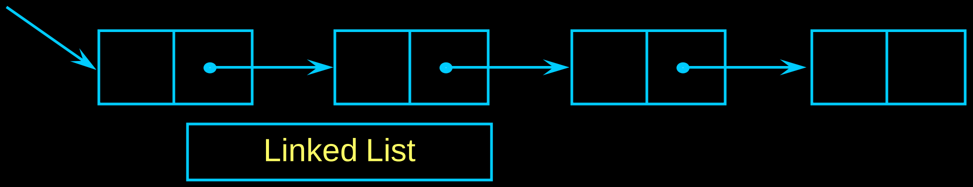
Truck

  - Conceptual entity



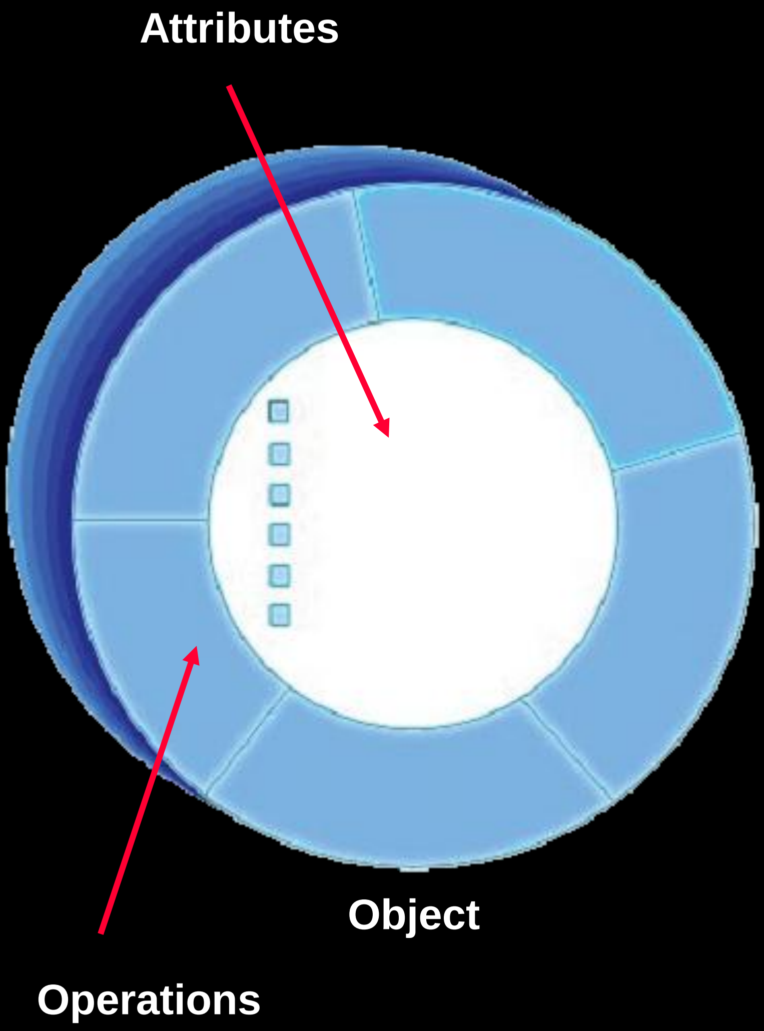
Chemical Process

  - Software entity



A More Formal Definition

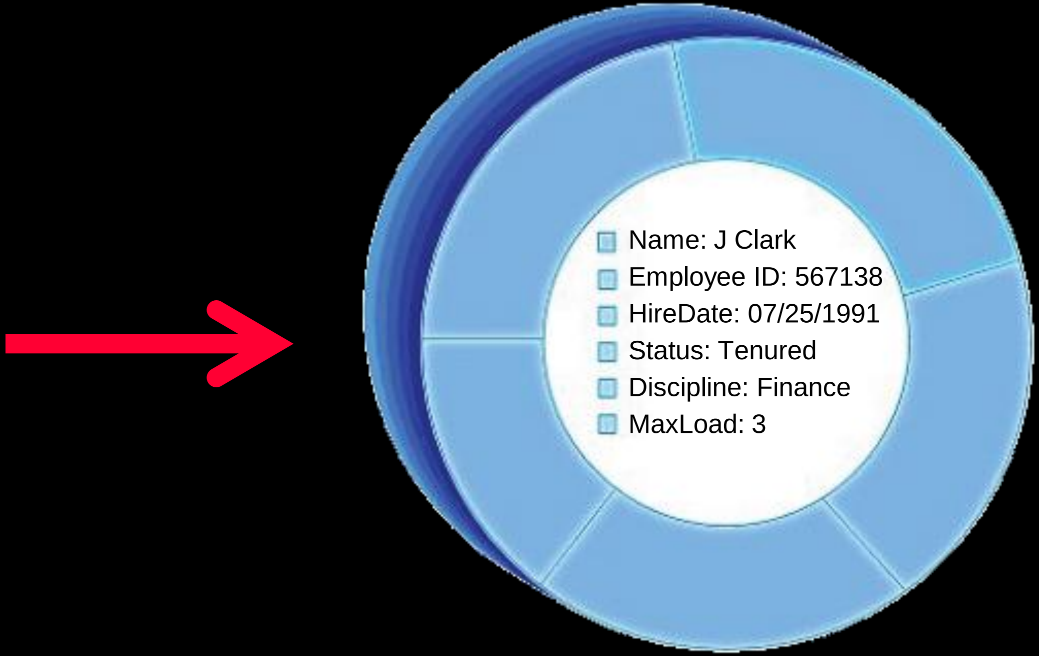
- An object is an entity with a well-defined boundary and identity that encapsulates *state* and *behavior.*
  - State is represented by attributes and relationships.
  - Behavior is represented by operations, methods, and state machines.



An Object Has State

- The state of an object is one of the possible conditions in which the object may exist.
- The state of an object normally changes over time.

Name: J Clark
Employee ID: 567138
Date Hired: July 25, 1991
Status: Tenured
Discipline: Finance
Maximum Course Load: 3 classes



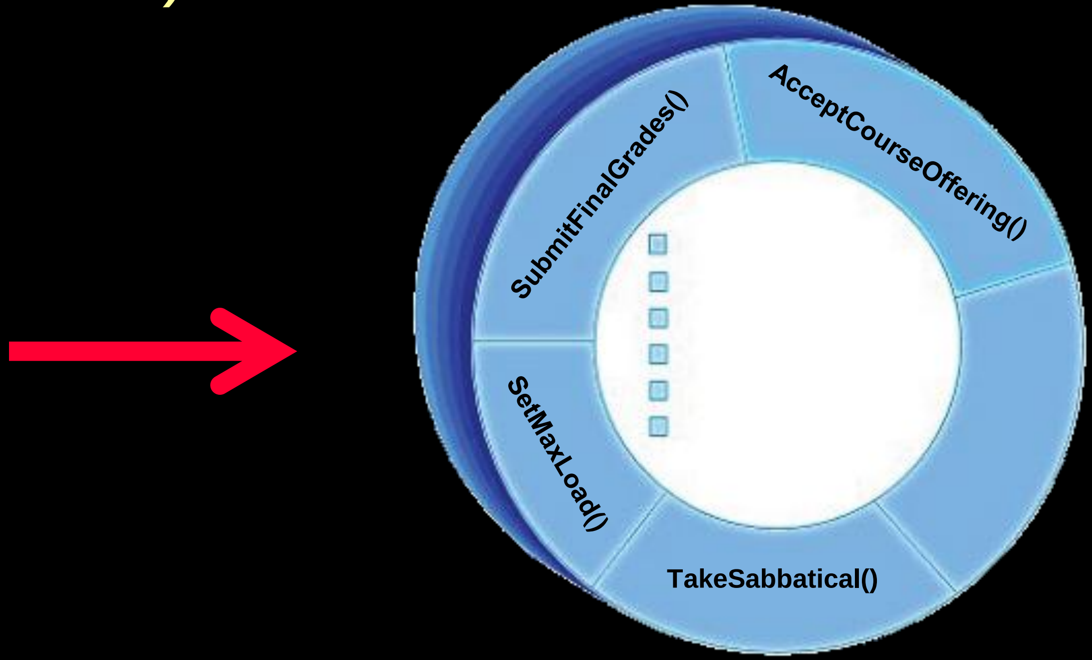
□ Name: J Clark
□ Employee ID: 567138
□ HireDate: 07/25/1991
□ Status: Tenured
□ Discipline: Finance
□ MaxLoad: 3

Professor Clark

An Object Has Behavior

- Behavior determines how an object acts and reacts.
- The visible behavior of an object is modeled by the set of messages it can respond to (operations the object can perform).

Professor Clark's behavior
Submit Final Grades
Accept Course Offering
Take Sabbatical
Maximum Course Load: 3 classes



Professor Clark

An Object Has Identity

- Each object has a unique identity, even if the state is identical to that of another object.

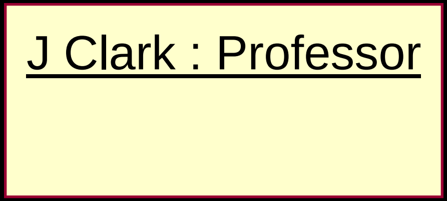
Professor “J Clark” teaches Biology

Professor “J Clark” teaches Biology

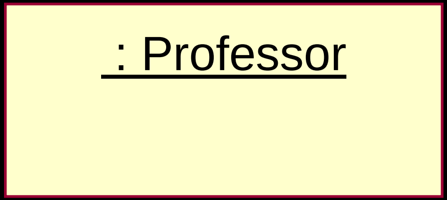
Representing Objects in the UML

- An object is represented as a rectangle with an underlined name.

Professor J Clark



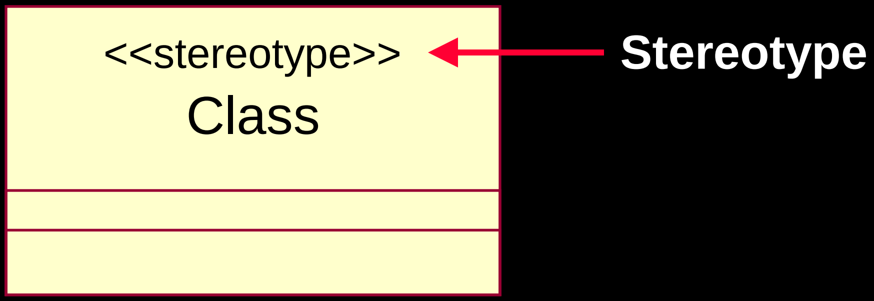
Named Object



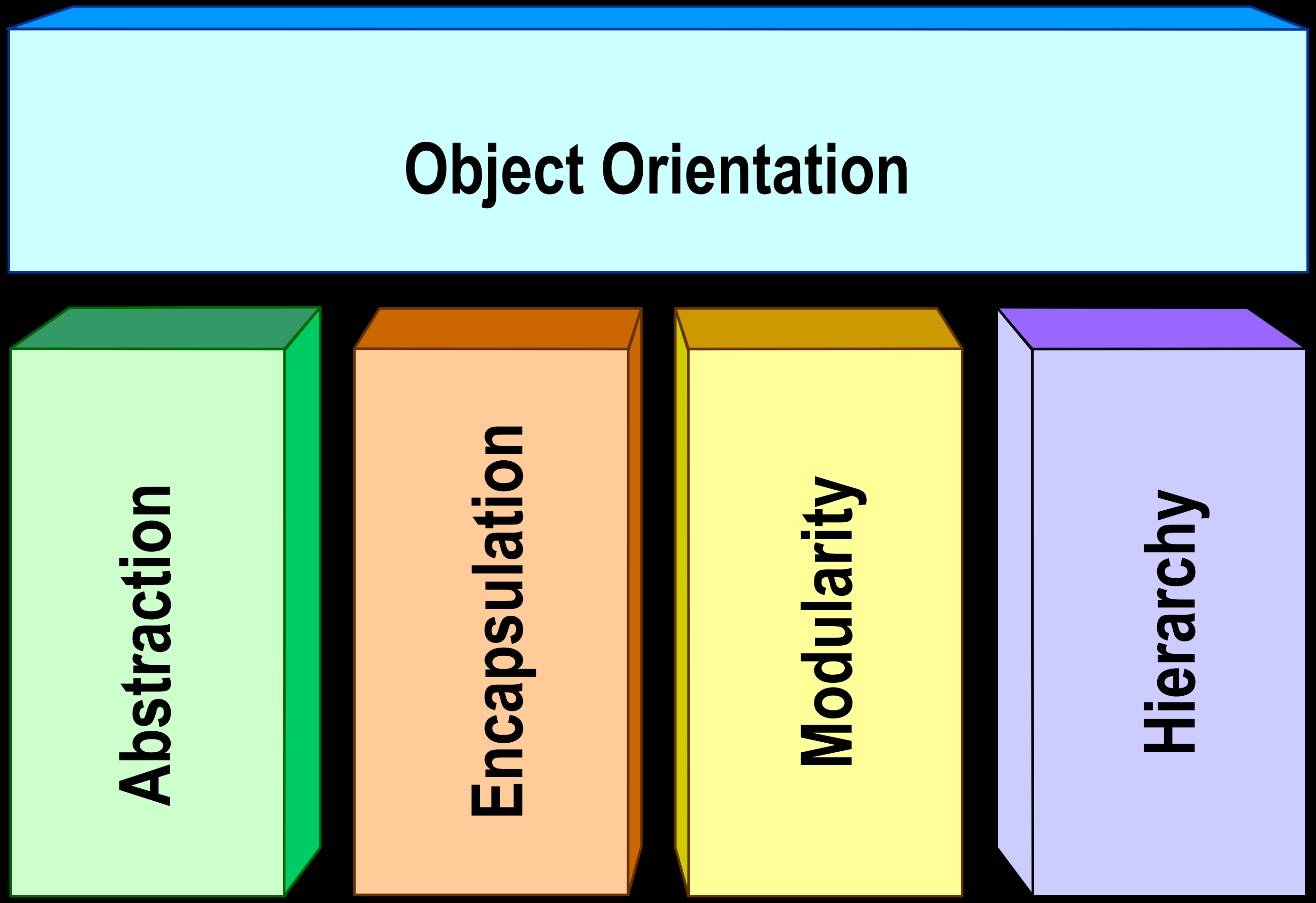
Unnamed Object

What Are Stereotypes?

- Stereotypes define a new model element in terms of another model element.
- Sometimes you need to introduce new things that speak the language of your domain and look like primitive building blocks.

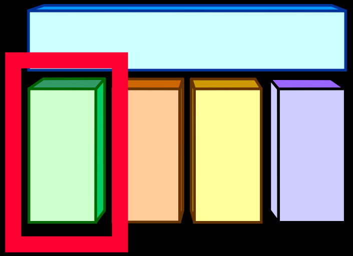


Basic Principles of Object Orientation

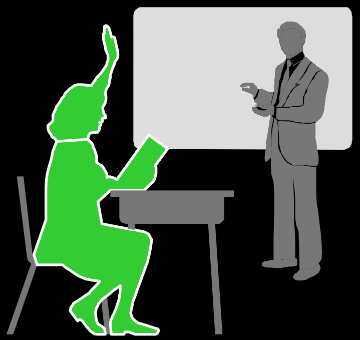


What Is Abstraction?

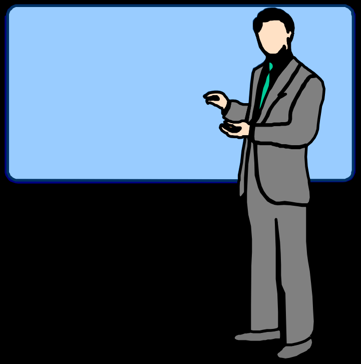
- The essential characteristics of an entity that distinguish it from all other kinds of entities
- Defines a boundary relative to the perspective of the viewer
- Is not a concrete manifestation, denotes the ideal essence of something



Example: Abstraction

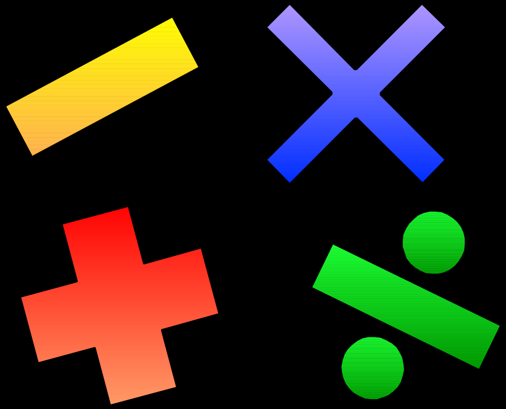


Student



Professor

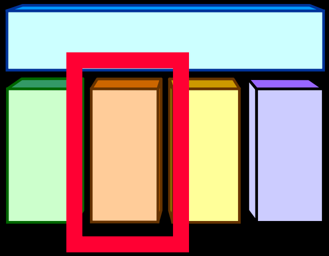
Course Offering (9:00 AM, Monday-Wednesday-Friday)

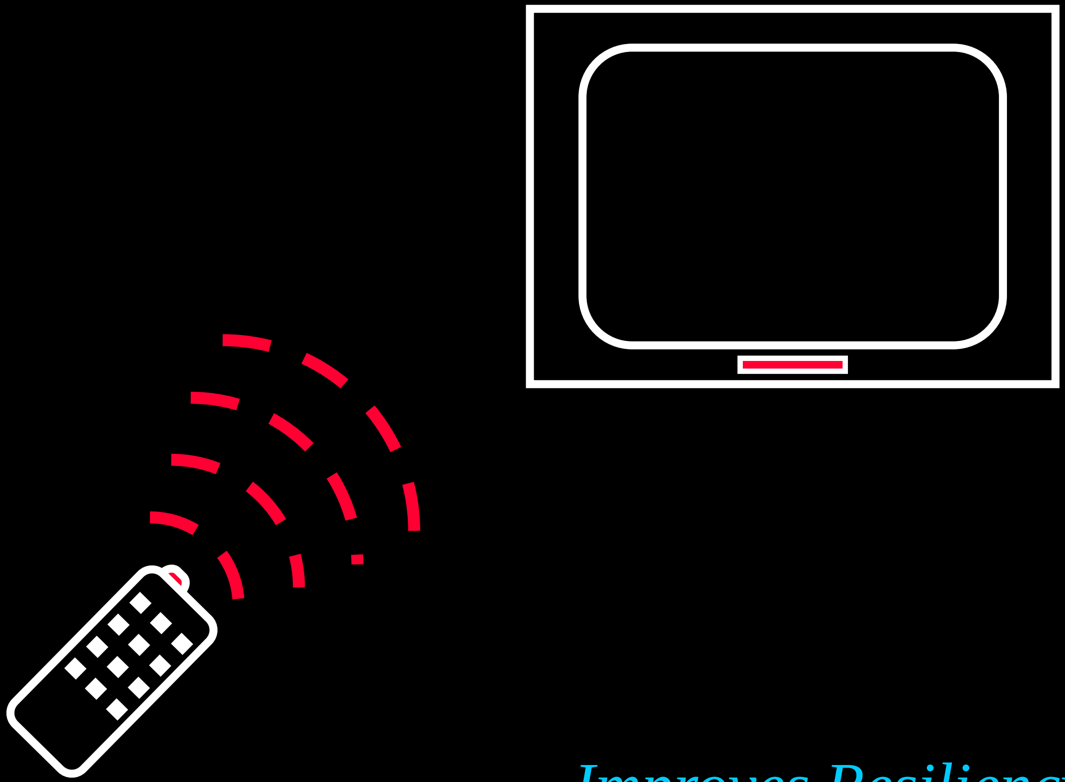


Course (e.g., Algebra)

What Is Encapsulation?

- Hide implementation from clients.
  - Clients depend on interface.

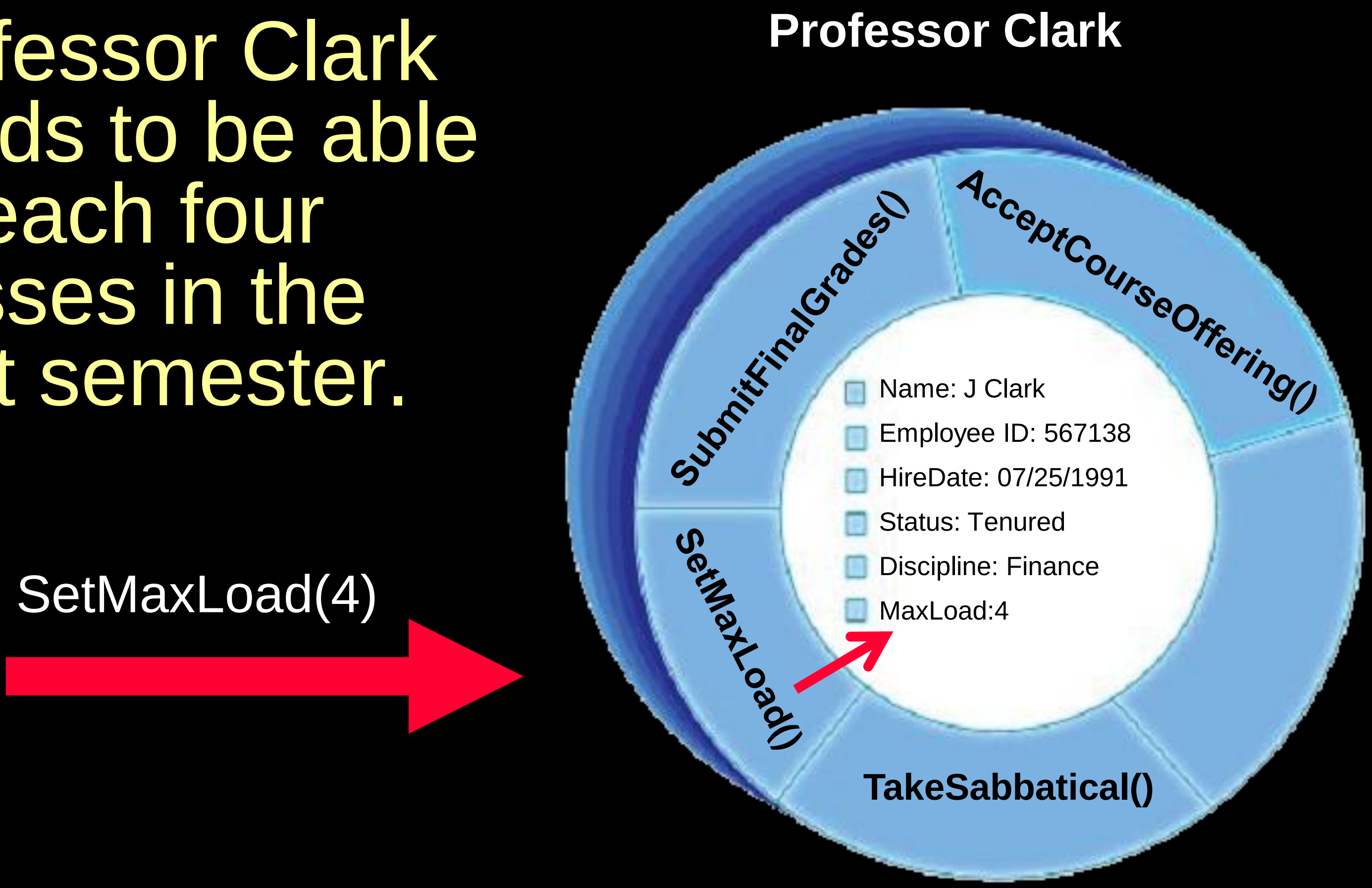




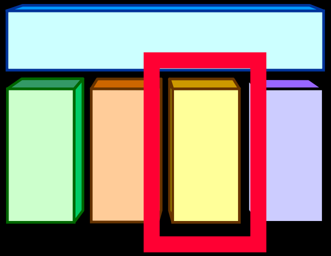
Improves Resiliency

Encapsulation Illustrated

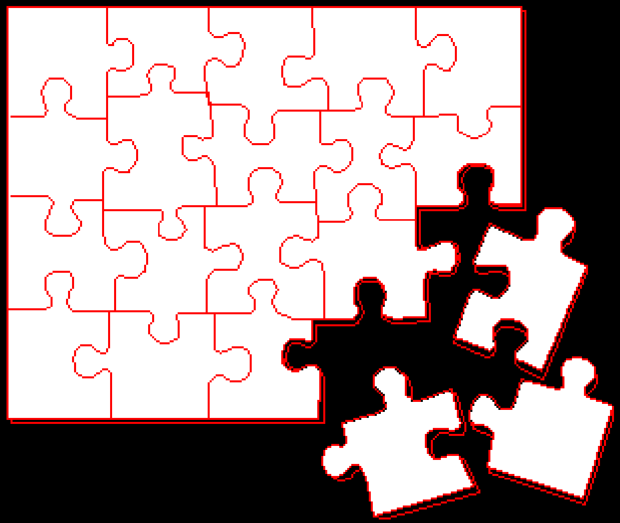
- Professor Clark needs to be able to teach four classes in the next semester.



What Is Modularity?

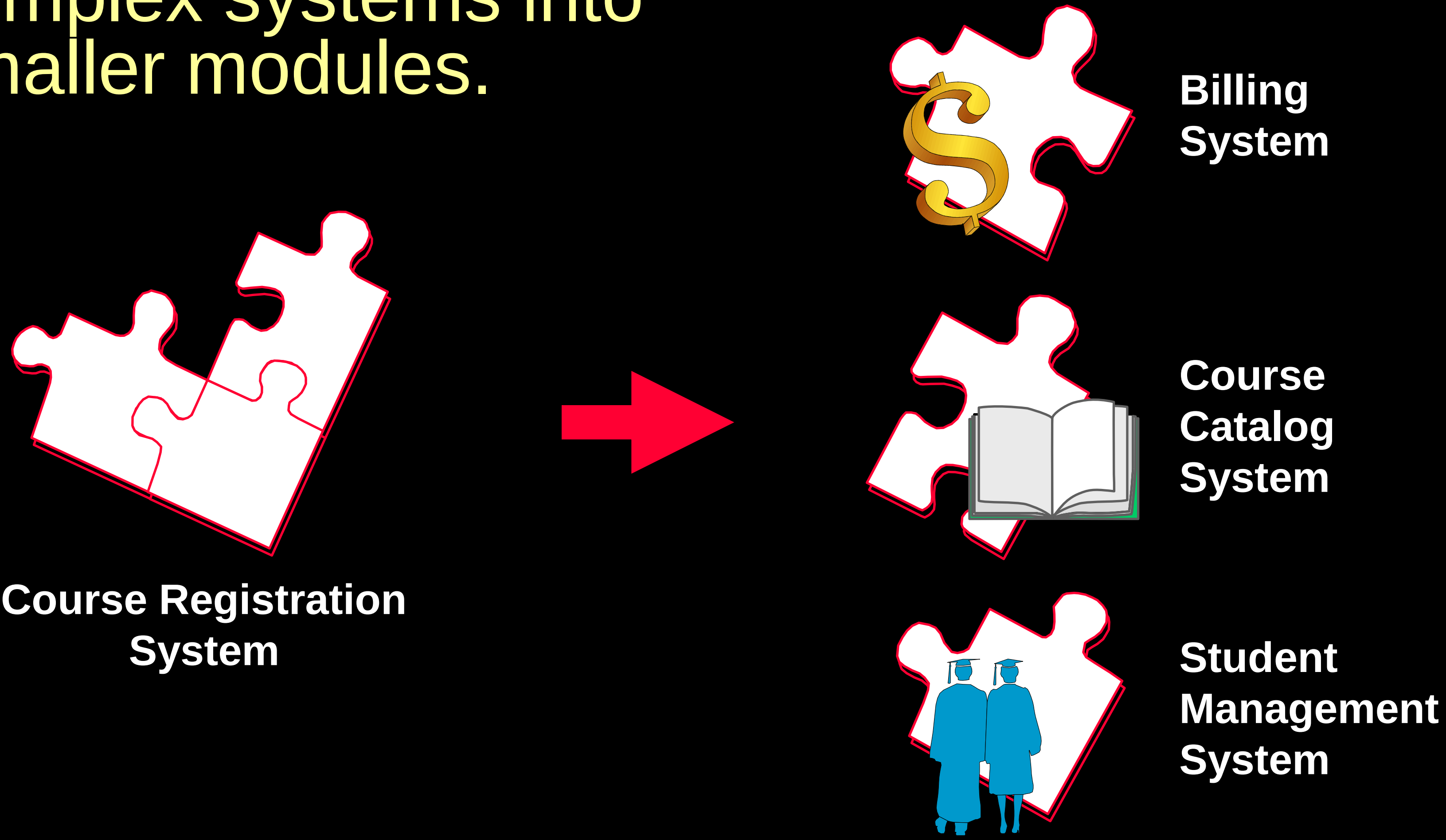


- Modularity is the breaking up of something complex into manageable pieces.
- Modularity helps people to understand complex systems.



Example: Modularity

- For example, break complex systems into smaller modules.



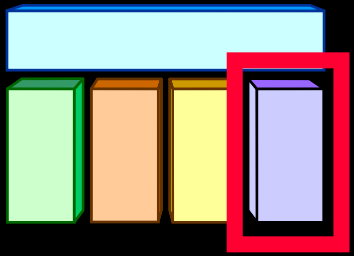
Course Registration System

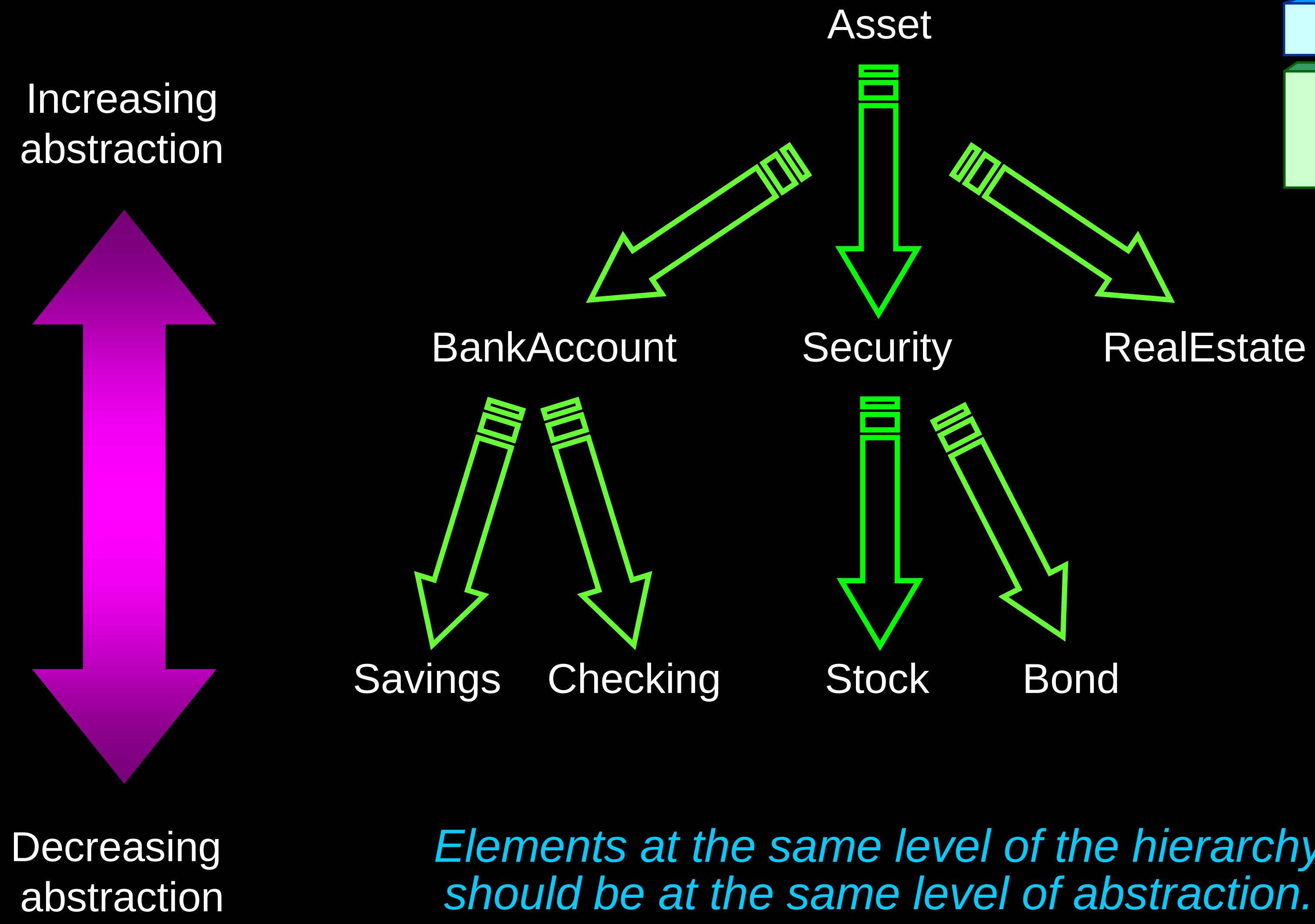
Billing System

Course Catalog System

Student Management System

What Is Hierarchy?

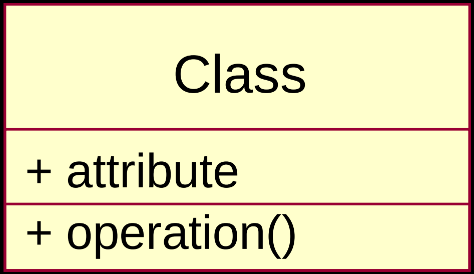




Elements at the same level of the hierarchy should be at the same level of abstraction.

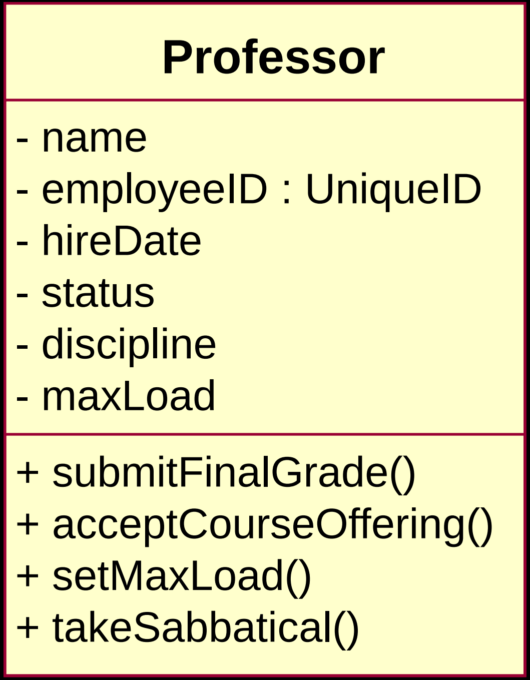
What Is a Class?

- A class is a description of a set of objects that share the same *attributes*, *operations*, *relationships*, and semantics.
  - An object is an instance of a class.
- A class is an abstraction in that it
  - Emphasizes relevant characteristics.
  - Suppresses other characteristics.



Representing Classes in the UML

- A class is represented using a rectangle with compartments.



Professor

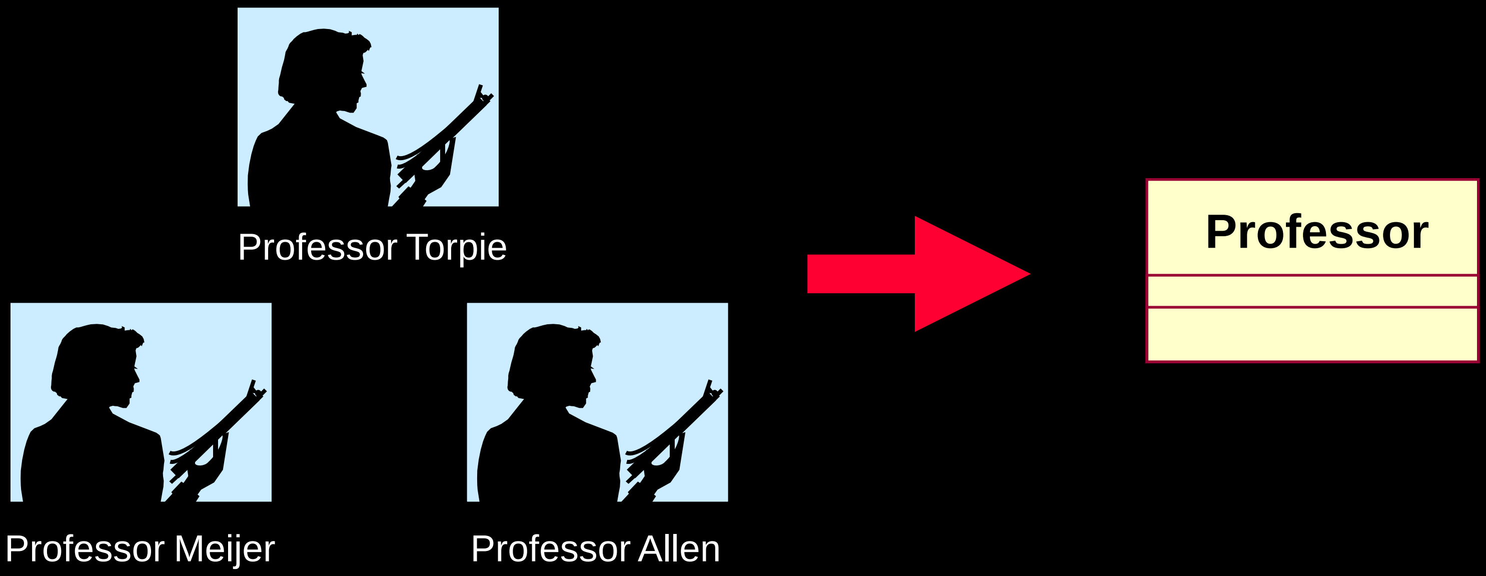
- name
- employeeID : UniqueID
- hireDate
- status
- discipline
- maxLoad

+ submitFinalGrade()
+ acceptCourseOffering()
+ setMaxLoad()
+ takeSabbatical()

Professor J Clark

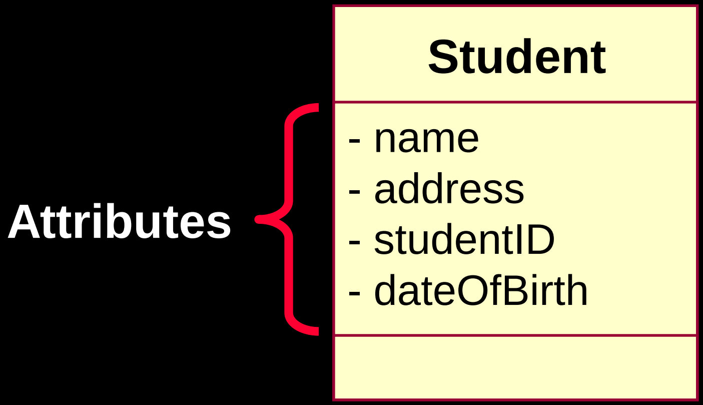
The Relationship Between Classes and Objects

- A class is an abstract definition of an object.
  - It defines the structure and behavior of each object in the class.
  - It serves as a template for creating objects.
- Classes are not collections of objects.



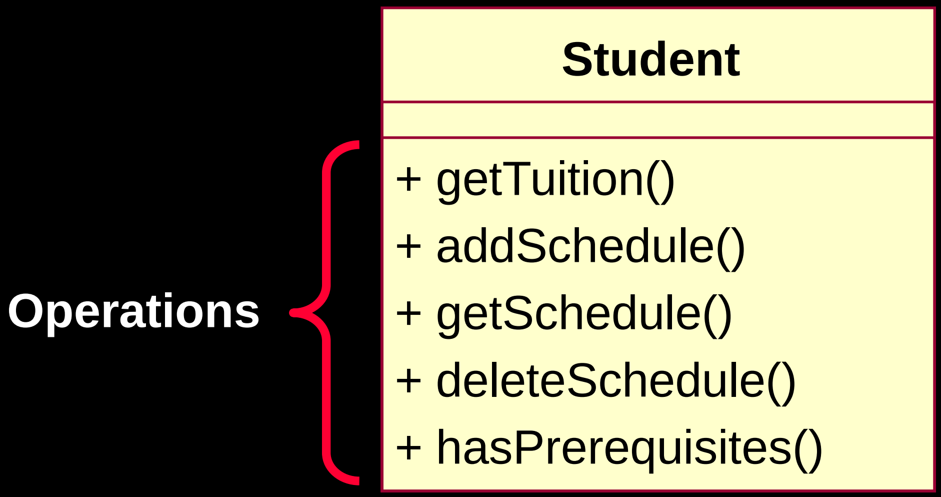
What Is an Attribute?

- An attribute is a named property of a class that describes a range of values that instances of the property may hold.
  - A class may have any number of attributes or no attributes at all.



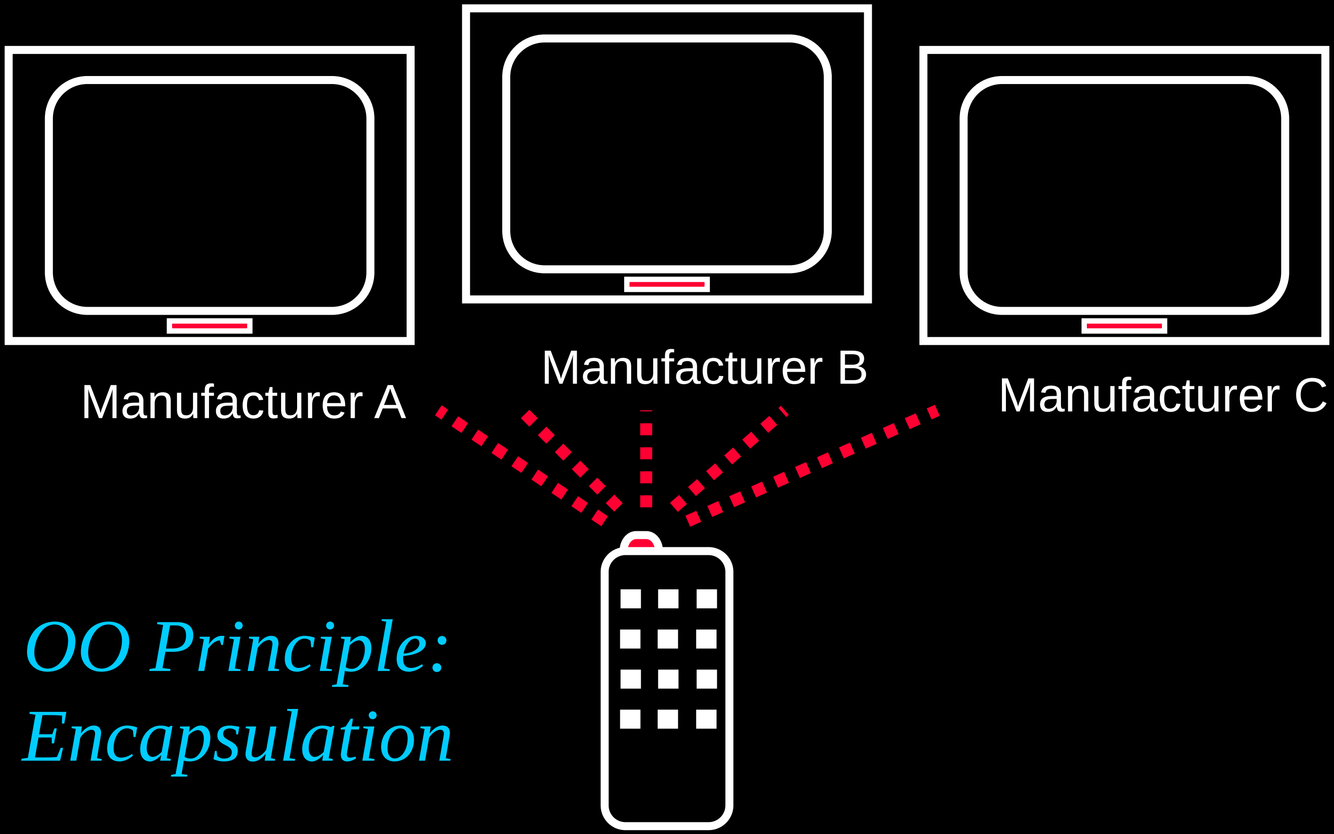
What Is an Operation?

- An operation is the implementation of a service that can be requested from any object of the class to affect behavior.
- A class may have any number of operations or none at all.



What Is Polymorphism?

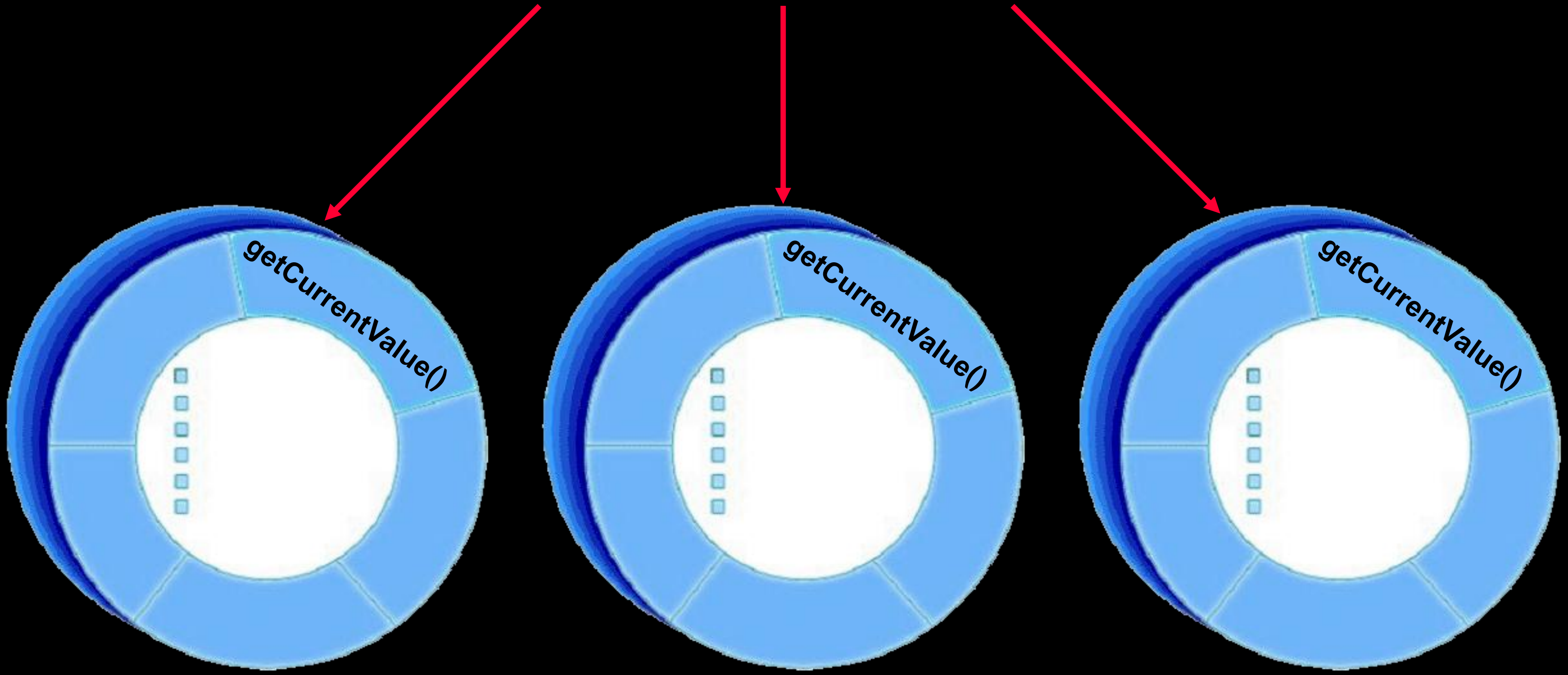
- The ability to hide many different implementations behind a single interface



OO Principle: Encapsulation

Example: Polymorphism

financialInstrument.getCurrentValue()



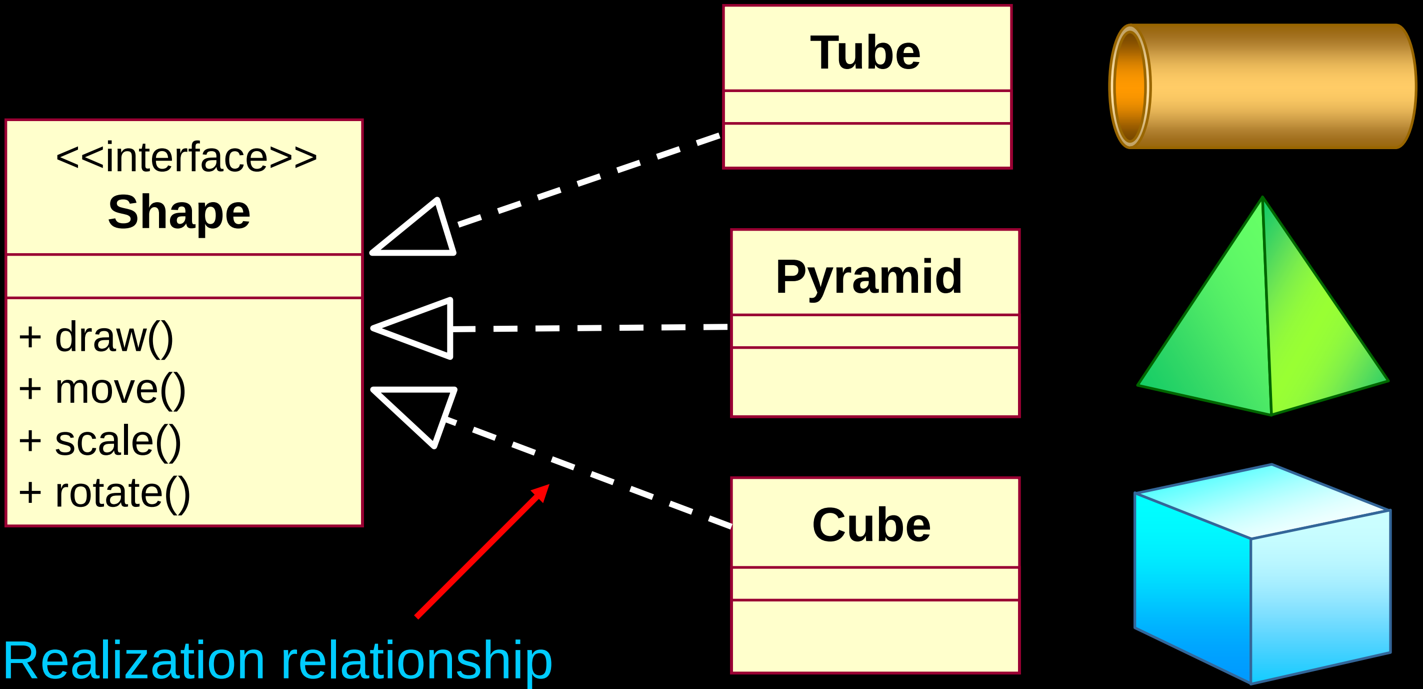
Stock

Bond

Mutual Fund

What Is an Interface?

- Interfaces formalize polymorphism
- Interfaces support “plug-and-play” architectures

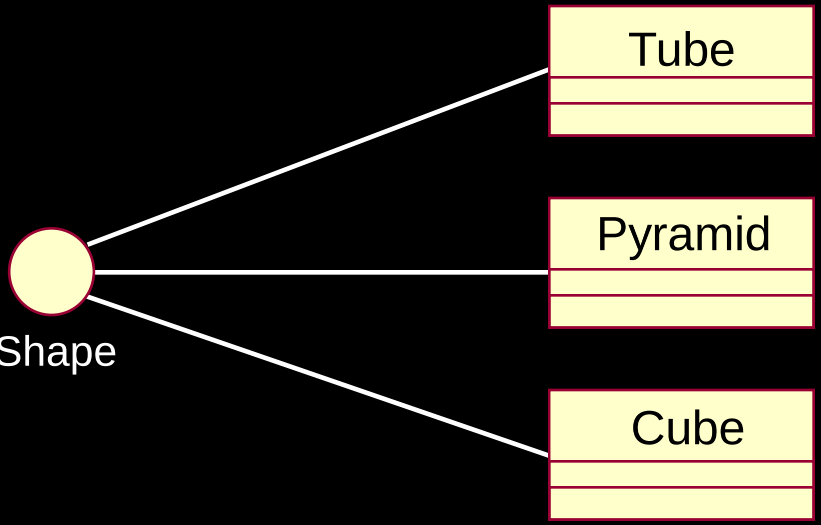


Realization relationship

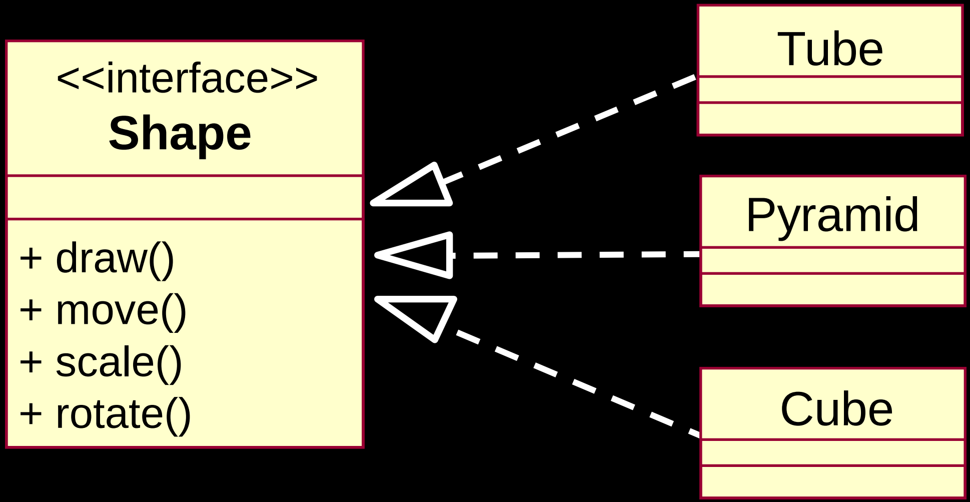
(stay tuned for realization relationships)

How Do You Represent an Interface?

Elided/Iconic Representation ("lollipop")

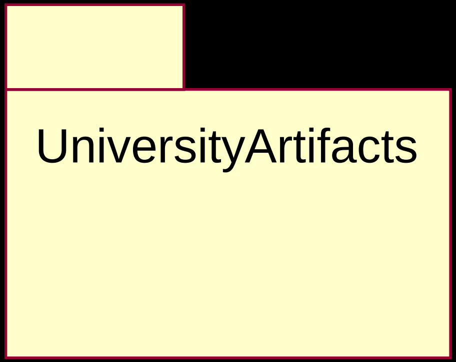


Canonical (Class/Stereotype) Representation



What Is a Package?

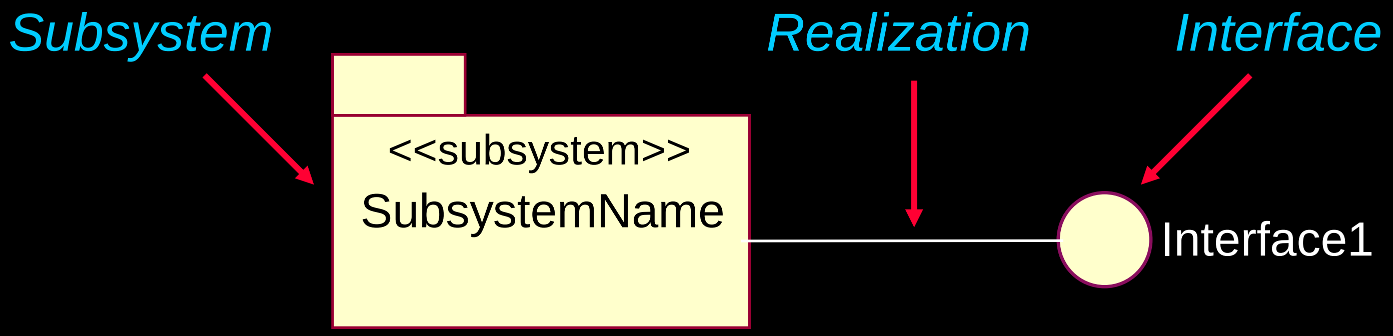
- A package is a general-purpose mechanism for organizing elements into groups.
- It is a model element that can contain other model elements.



- A package can be used:
  - To organize the model under development.
  - As a unit of configuration management.

What Is a Subsystem?

- A combination of a package (can contain other model elements) and a class (has behavior)
- Realizes one or more interfaces which define its behavior

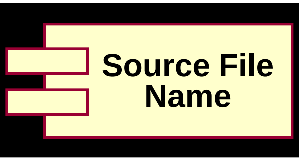
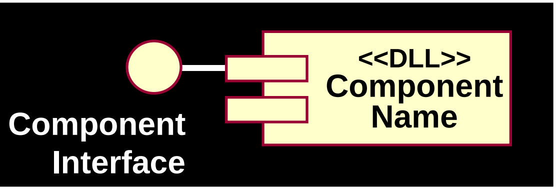
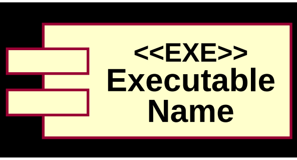


OO Principles: Encapsulation and Modularity

(stay tuned for realization relationships)

What Is a Component?

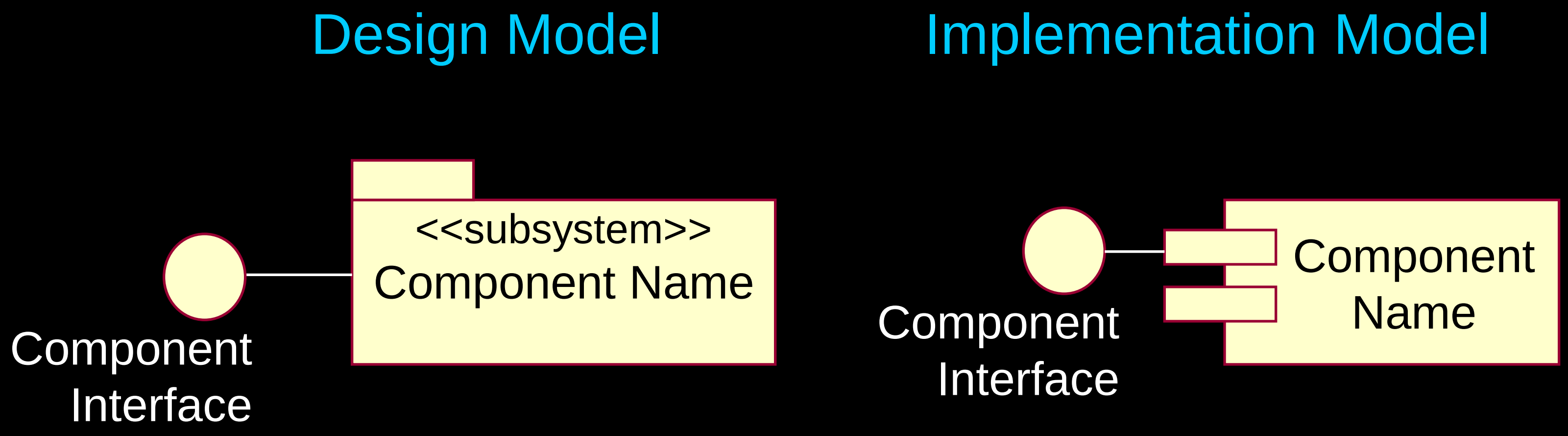
- A non-trivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture
- A component may be

▪ A source code component	
▪ A run time component	
▪ An executable component	

OO Principle: Encapsulation

Subsystems and Components

- Components are the physical realization of an abstraction in the design
- Subsystems can be used to represent the component in the design



Design Model

Implementation Model

<<subsystem>>
Component Name

Component Interface

Component Name

Component Interface

OO Principles: Encapsulation and Modularity

Components UML 1.4 and Beyond

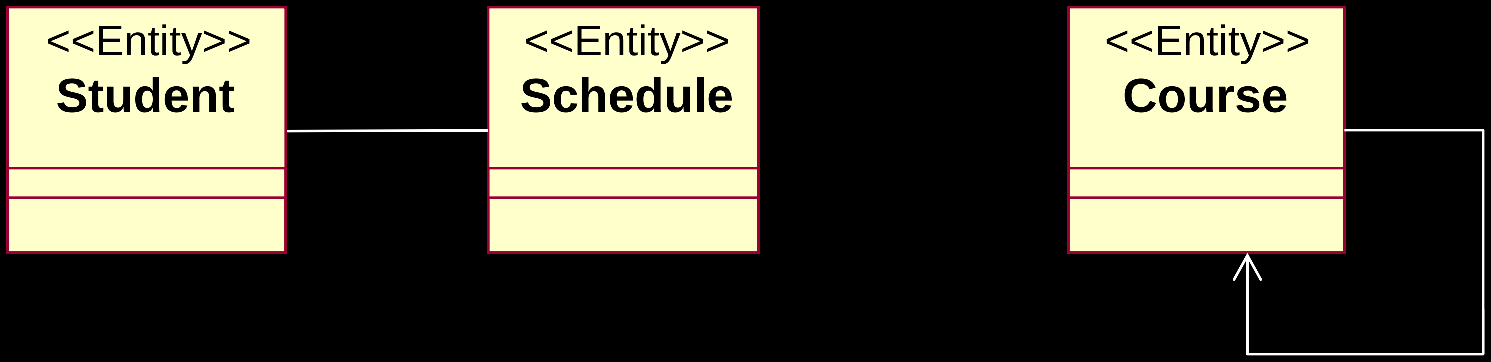
- UML 1.4 introduces concept of Artifacts:
  - An Artifact represents a physical piece of information that is used or produced by a software development process. Examples of Artifacts include models, source files, scripts, and binary executable files.
- To distinguish between artifacts in general, and the artifacts that make up the implementation, we introduce a new term:
  - Implementation Element - the physical parts (UML artifacts) that make up an implementation, including software code files (source, binary or executable), and data files.

Components UML 1.4 and Beyond (cont.)

- Component becomes similar to subsystem:
  - can group classes to define a larger granularity units of a system
  - can separate the visible interfaces from internal implementation
  - can have instances that execute at run-time
- The distinction between "component" and "artifact" is new in UML 1.4.
  - Many tools, profiles, and examples continue to use "component" to represent implementation elements.

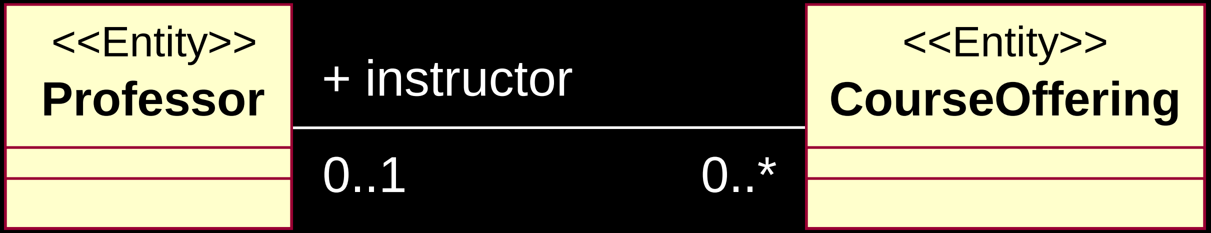
What Is an Association?

- The semantic relationship between two or more classifiers that specifies connections among their instances
  - A structural relationship, specifying that objects of one thing are connected to objects of another



What Is Multiplicity?

- Multiplicity is the number of instances one class relates to ONE instance of another class.
- For each association, there are two multiplicity decisions to make, one for each end of the association.
  - For each instance of Professor, many Course Offerings may be taught.
  - For each instance of Course Offering, there may be either one or zero Professor as the instructor.

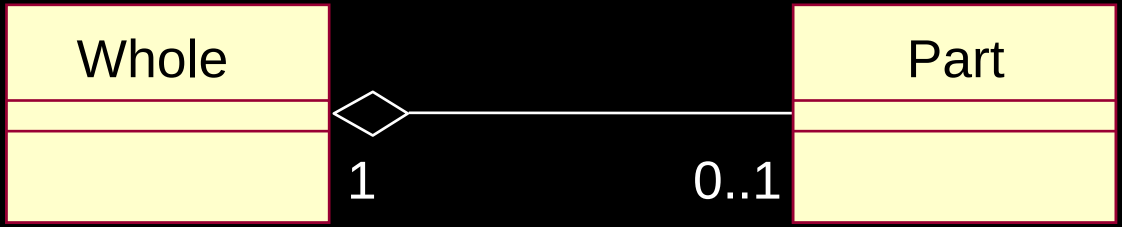


Multiplicity Indicators

Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or One (optional scalar role)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6

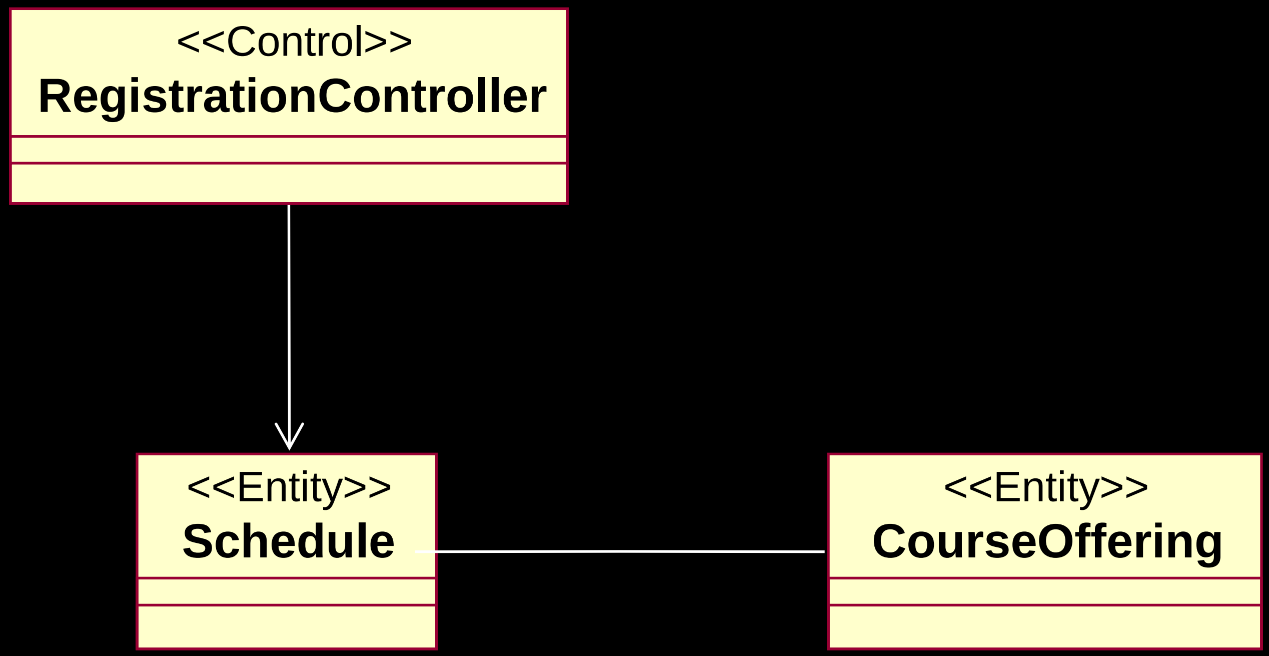
What Is Aggregation?

- An aggregation is a special form of association that models a whole-part relationship between an aggregate (the whole) and its parts.
  - An aggregation is an “Is a part-of” relationship.
- Multiplicity is represented like other associations.



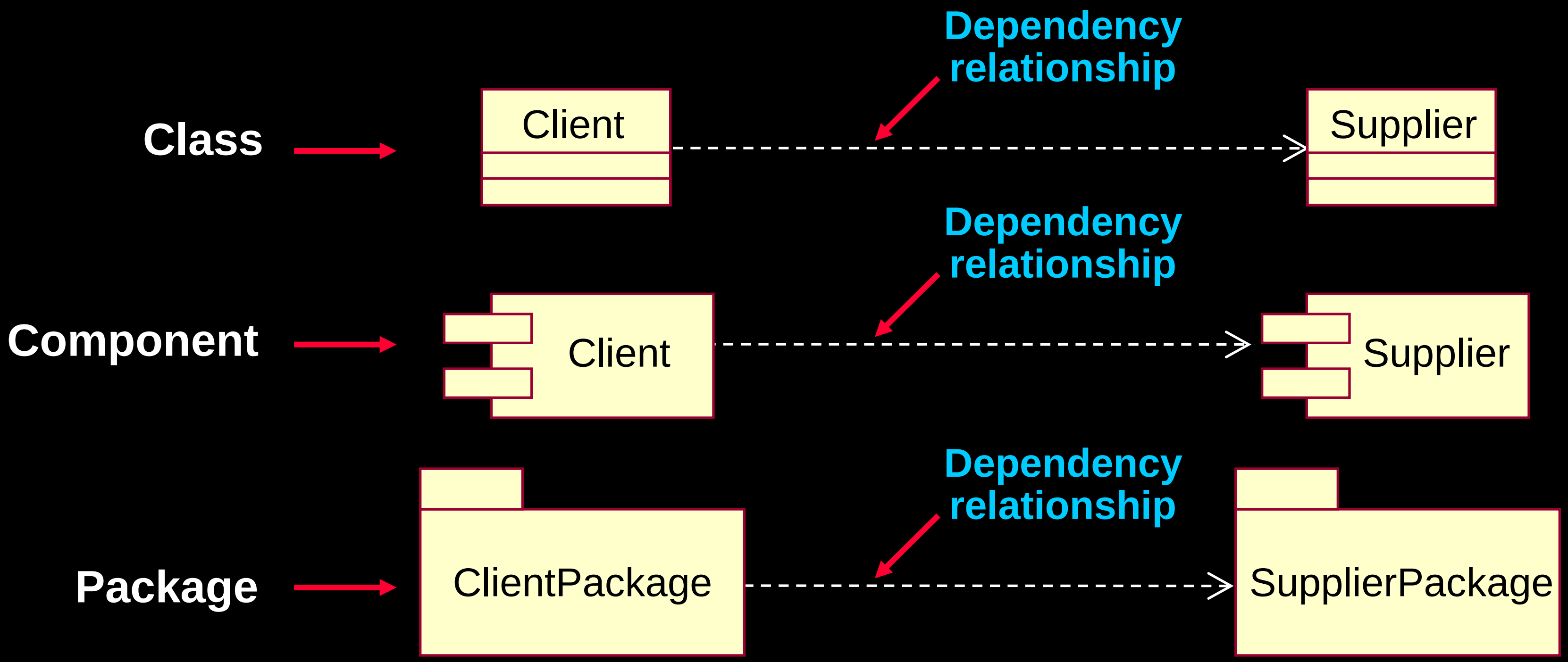
What Is Navigability?

- Indicates that it is possible to navigate from a associating class to the target class using the association



Relationships: Dependency

- A relationship between two model elements where a change in one may cause a change in the other
- Non-structural, “using” relationship

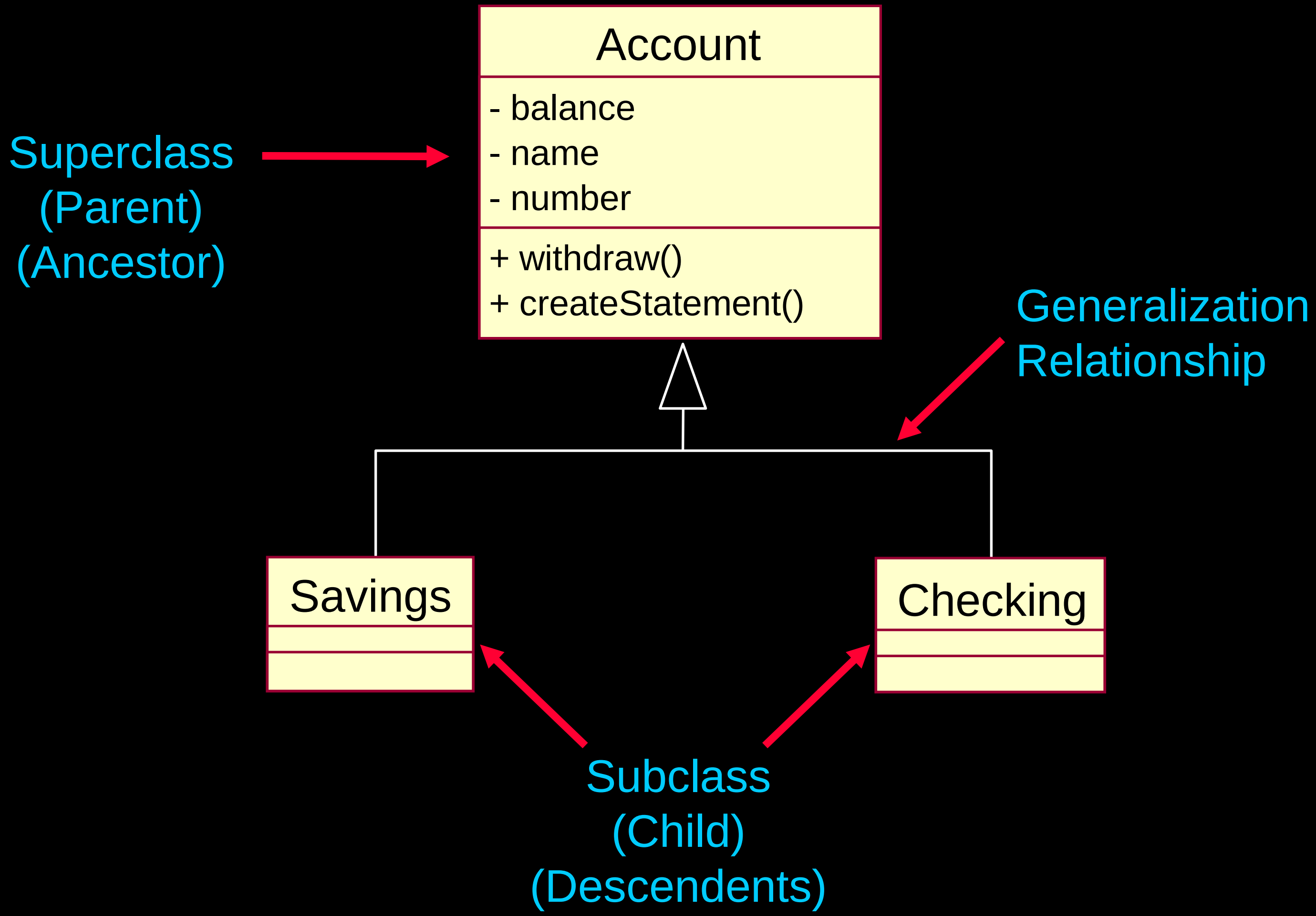


What Is Generalization?

- A relationship among classes where one class shares the structure and/or behavior of one or more classes
- Defines a hierarchy of abstractions in which a subclass inherits from one or more superclasses
  - Single inheritance
  - Multiple inheritance
- Is an “is a kind of” relationship

Example: Single Inheritance

- One class inherits from another



Superclass
(Parent)
(Ancestor)

Account

- balance
- name
- number

+ withdraw()
+ createStatement()

Generalization
Relationship

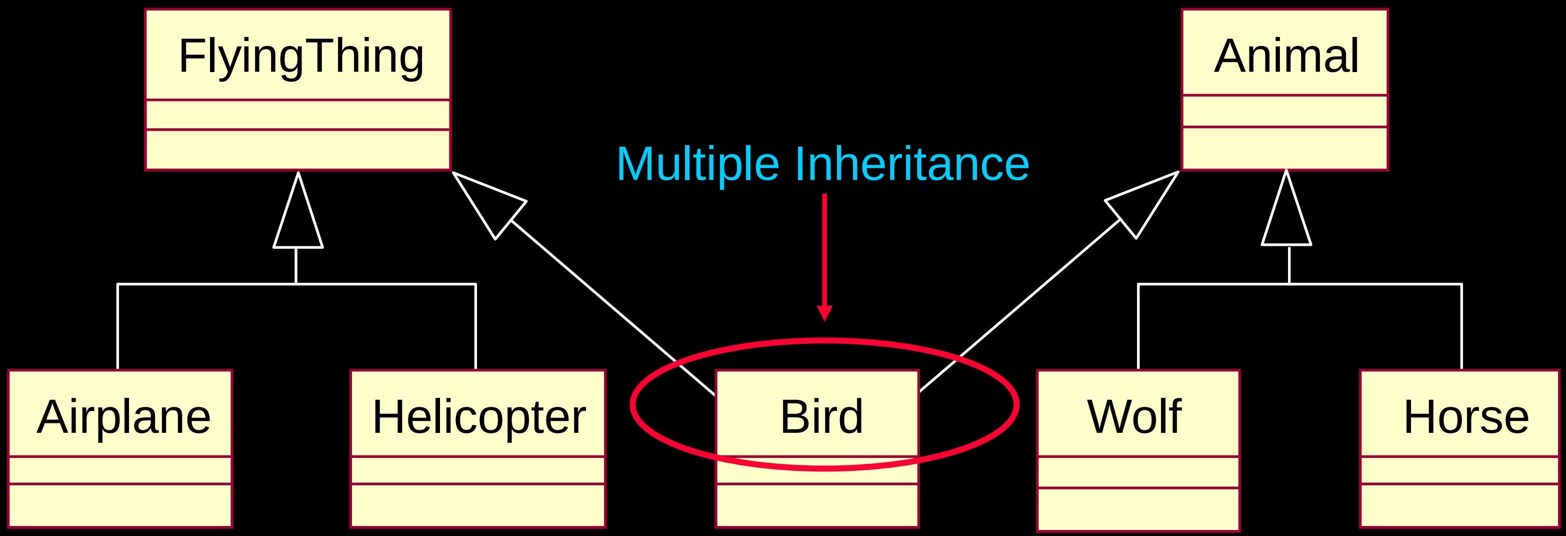
Savings

Checking

Subclass
(Child)
(Descendents)

Example: Multiple Inheritance

- A class can inherit from several other classes.



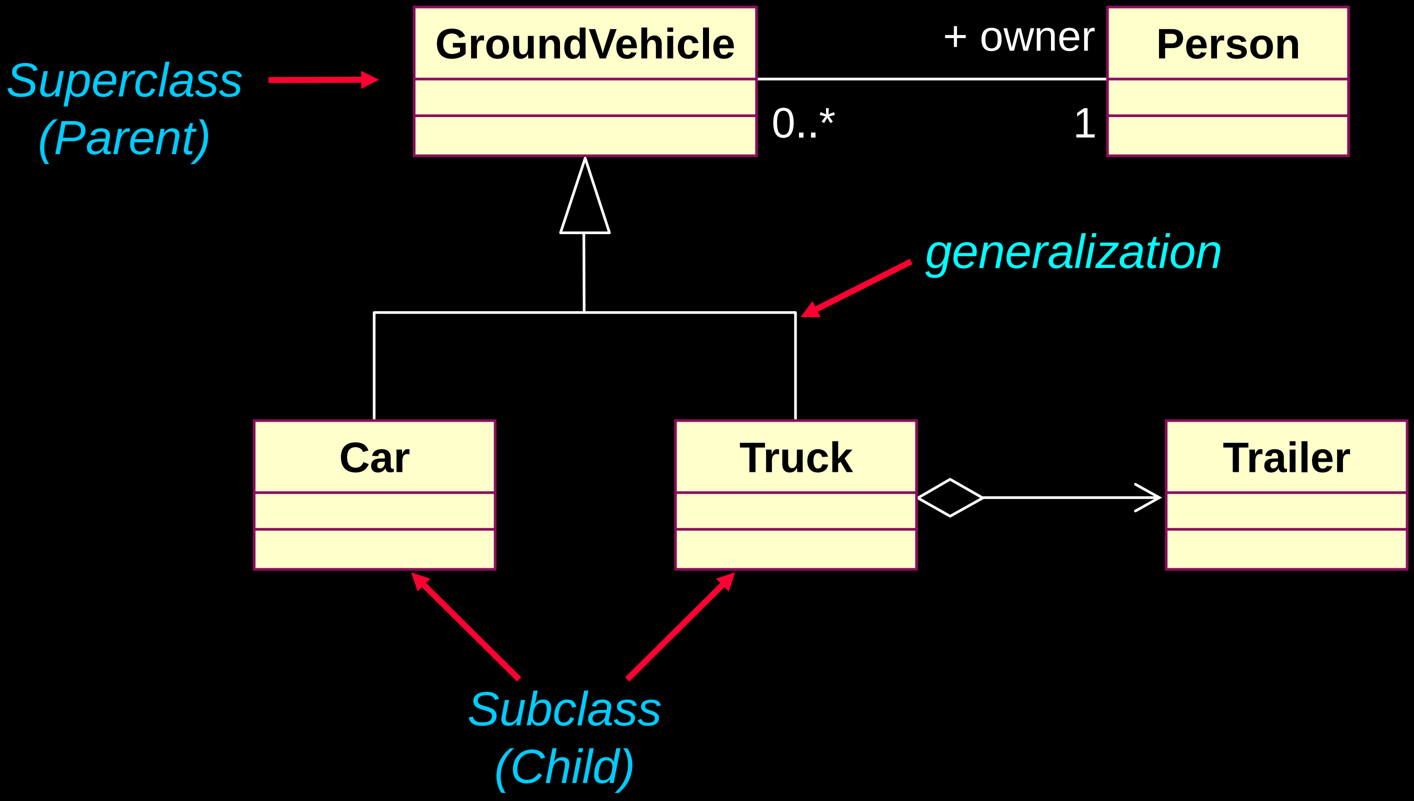
Use multiple inheritance only when needed and always with caution!

What Gets Inherited?

- A subclass inherits its parent's attributes, operations, and relationships
- A subclass may:
  - Add additional attributes, operations, relationships
  - Redefine inherited operations (use caution!)
- Common attributes, operations, and/or relationships are shown at the highest applicable level in the hierarchy

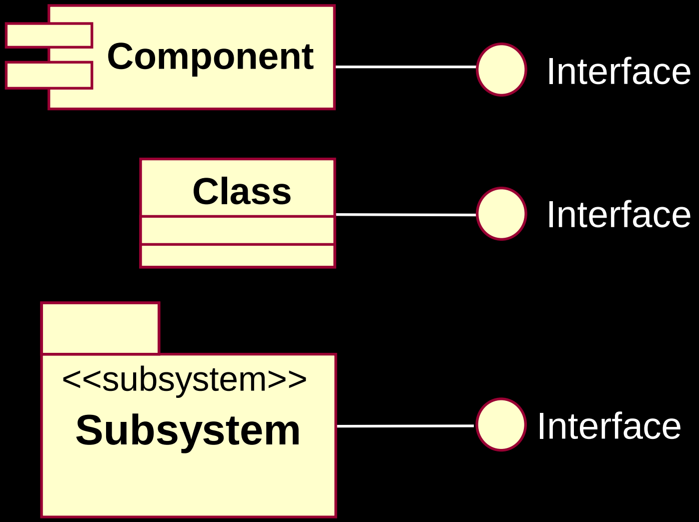
Inheritance leverages the similarities among classes

Example: What Gets Inherited

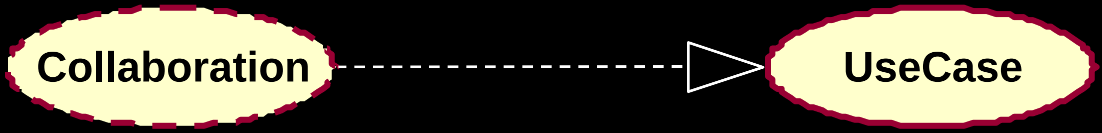


What Is Realization?

- One classifier serves as the contract that the other classifier agrees to carry out, found between:
  - Interfaces and the classifiers that realize them

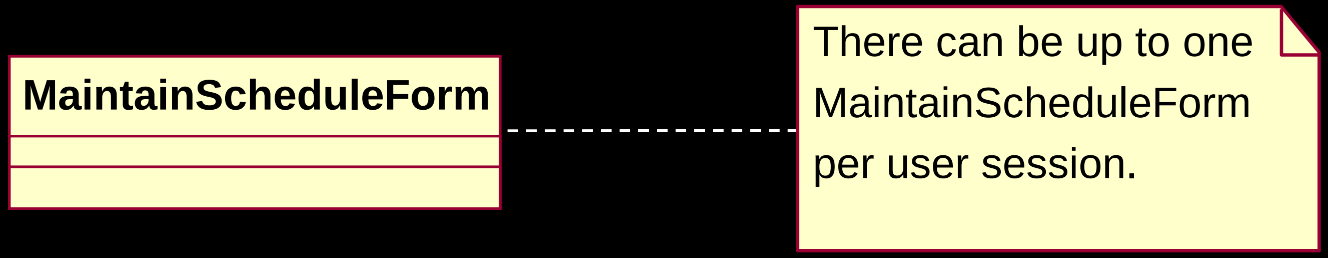


  - Use cases and the collaborations that realize them



What Are Notes?

- A comment that can be added to include more information on the diagram
- May be added to any UML element
- A “dog eared” rectangle
- May be anchored to an element with a dashed line


MaintainScheduleForm
There can be up to one MaintainScheduleForm per user session.

Review: Concepts of Object Orientation

- What are the four basic principles of object orientation? Provide a brief description of each.
- What is an object and what is a class? What is the difference between the two?
- What is an attribute?
- What is an operation?
- What is polymorphism? What is an interface?

Review: Concepts of Object Orientation (cont.)

- What is a package?
- What is a subsystem? How does it relate to a package? How does it relate to a class?
- Name the four basic UML relationships and describe each.
- Describe the strengths of object orientation
- What are stereotypes?