Object Oriented Analysis and Design (OOAD)

Introduction to Object Orientation

Source: Rational

Objectives: Introduction to Object Orientation

- Understand the basic principles of object orientation
- Understand the basic concepts and terms of object orientation and the associated UML notation
- Appreciate the strengths of object orientation

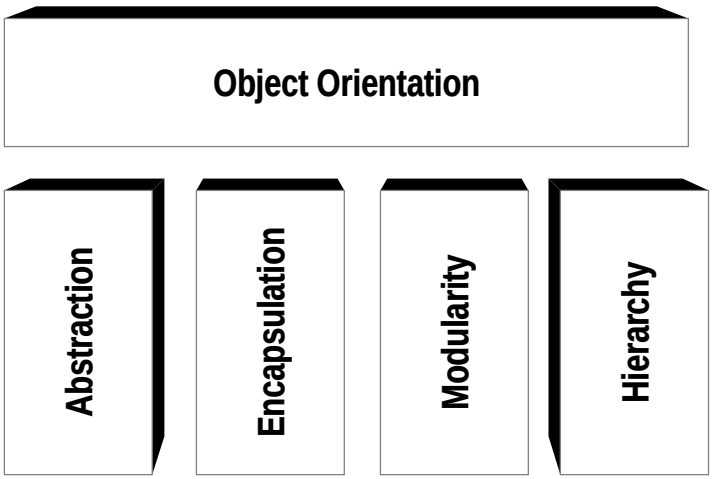
Source: Rational

Introduction to Object Orientation Topics

- Basic Principles of Object Orientation
- Basic Concepts of Object Orientation
- Strengths of Object Orientation

Source: Rational

Basic Principles of Object Orientation



Source: Rational

What is Abstraction?

Salesperson

Not saying Which salesperson – just a salesperson in general!!!

Customer

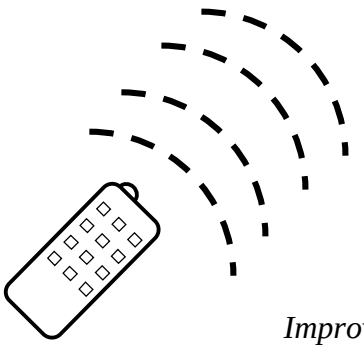
Product

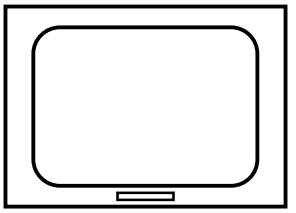
Manages Complexity

Source: Rational

What is Encapsulation?

- Hide implementation from clients
  - Clients depend on interface





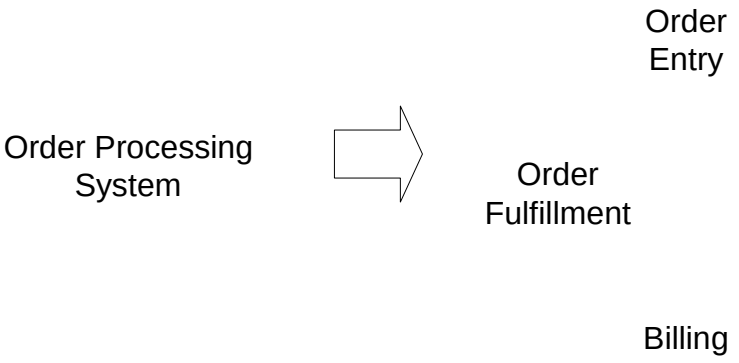
How does an object encapsulate?
What does it encapsulate?

Improves Resiliency

Source: Rational

What is Modularity?

- The breaking up of something complex into manageable pieces

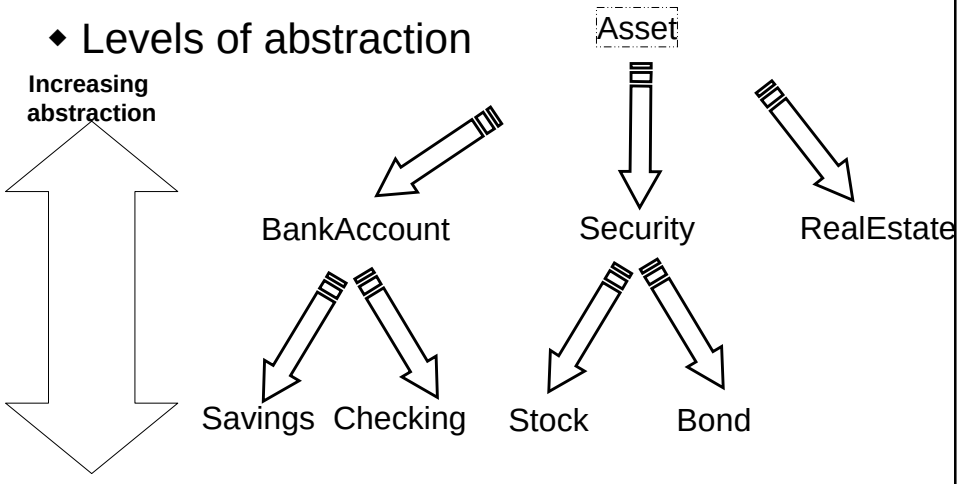


Manages Complexity

Source: Rational

What is Hierarchy?

- Levels of abstraction



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

Source: Rational

Introduction to Object Orientation Topics

- Basic Principles of Object Orientation
- Basic Concepts of Object Orientation
- Strengths of Object Orientation

Source: Rational

Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

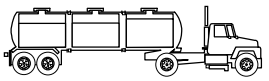
Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

What is an Object?

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

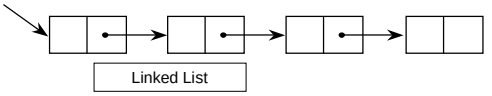


Truck

  - Conceptual entity

Chemical Process

  - Software entity



Source: Rational

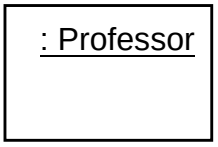
A More Formal Definition

- An object is a concept, abstraction, or thing with sharp boundaries and meaning for an application
- An object is something that has:
  - State
  - Behavior
  - Identity

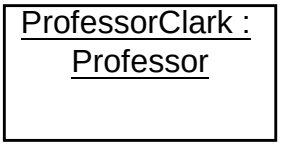
Source: Rational

Representing Objects

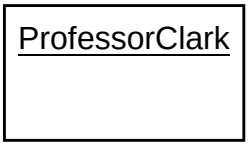
- An object is represented as rectangles with underlined names



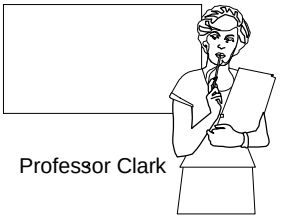
Class Name Only



Class and Object Name



Object Name Only



(stay tuned for classes)

Source: Rational

Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

What is a Class?

- A class is a description of a group of objects with common properties (attributes), behavior (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

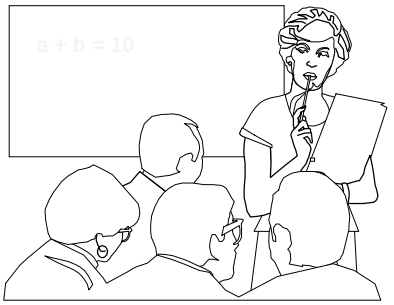
OO Principle: Abstraction

Source: Rational

Sample Class

Class
Course

Properties
Name
Location
Days offered
Credit hours
Start time
End time



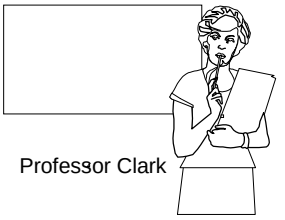
Behavior
Add a student
Delete a student
Get course roster
Determine if it is full

Source: Rational

Representing Classes

- A class is represented using a compartmented rectangle

Professor

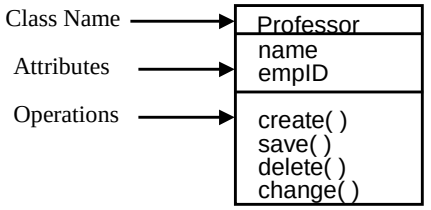


Professor Clark

Source: Rational

Class Compartments

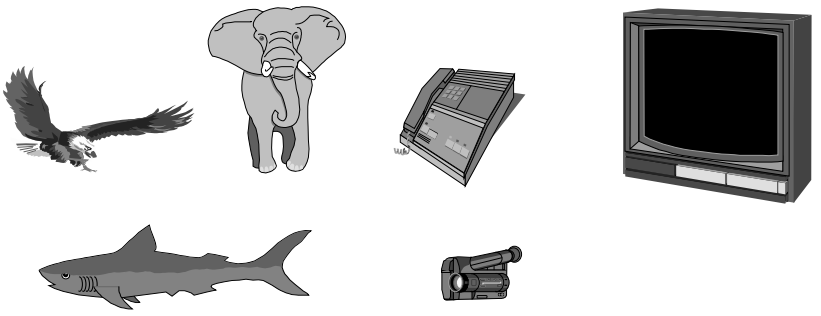
- A class is comprised of three sections
  - The first section contains the class name
  - The second section shows the structure (attributes)
  - The third section shows the behavior (operations)



Source: Rational

Classes of Objects

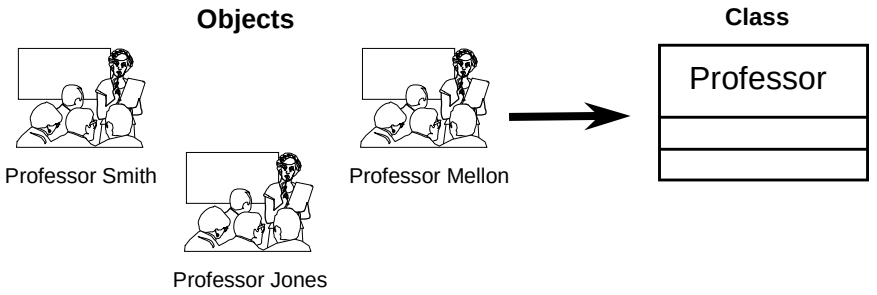
- How many classes do you see?



Source: Rational

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
- Objects are grouped into classes



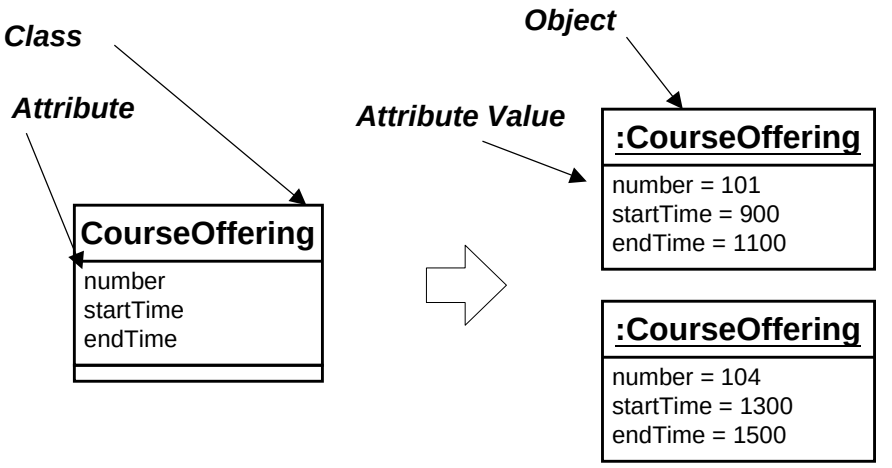
Source: Rational

Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

What is an Attribute?



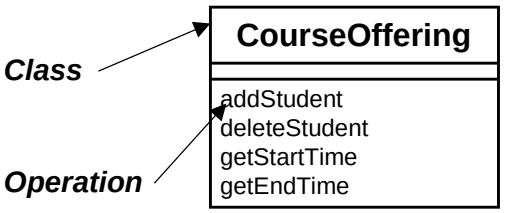
Source: Rational

Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

What is an Operation?



Source: Rational

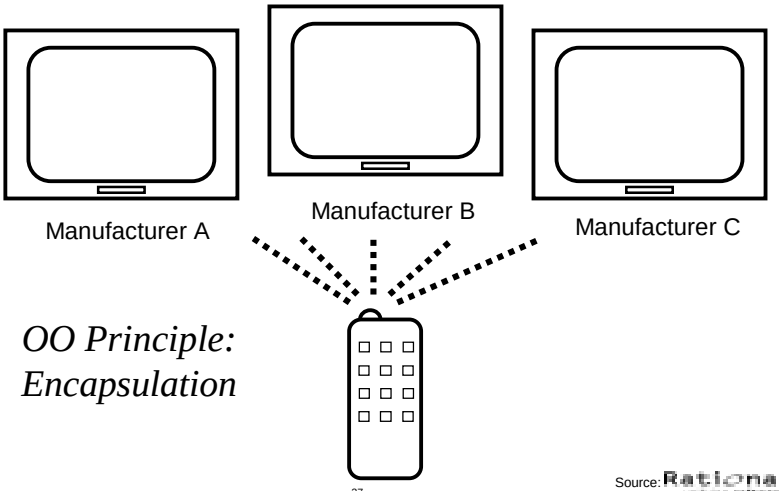
Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

What is Polymorphism?

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

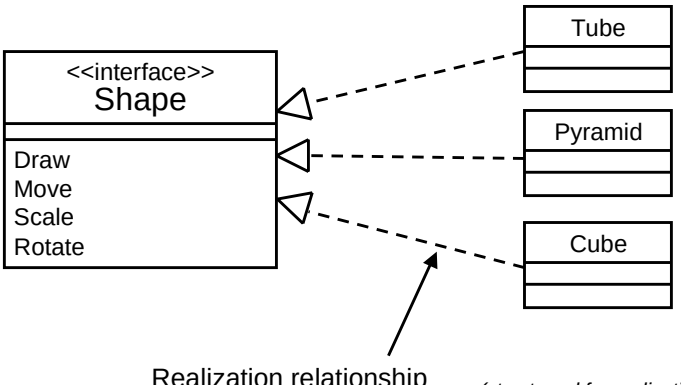


OO Principle: Encapsulation

Source: Rational

What is an Interface?

- Interfaces formalize polymorphism
- Interfaces support "plug-and-play" architectures

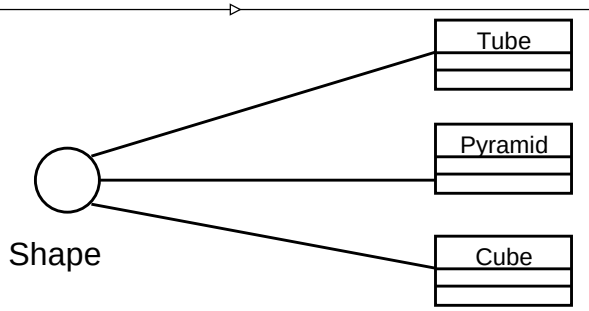


(stay tuned for realization relationships)

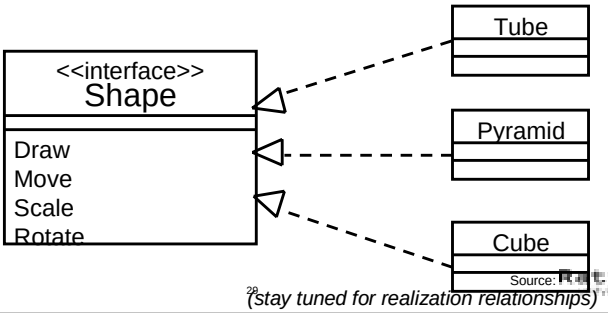
Source: Rational

Interface Representations

Elided/Iconic Representation ("lollipop")



Canonical (Class/Stereotype) Representation



Source: Rational

(stay tuned for realization relationships)

Basic Concepts of Object Orientation

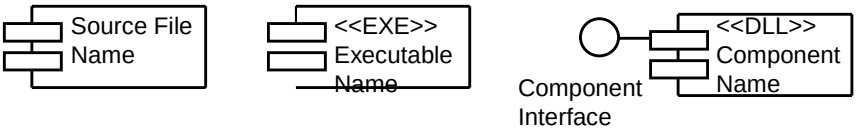
- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

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 components or
  - An executable component

OO Principle: Encapsulation



Source: Rational

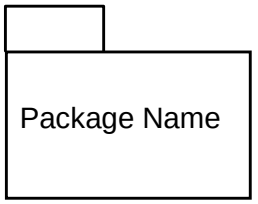
Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

What is a Package?

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



OO Principle: Modularity

- Uses
  - Organize the model under development
  - A unit of configuration management

Source: Rational

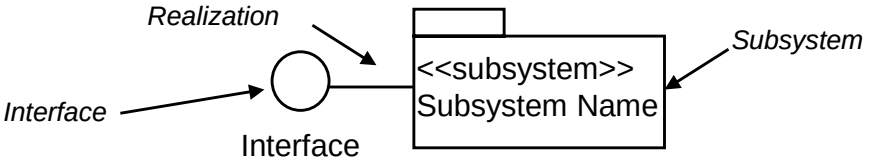
Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

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



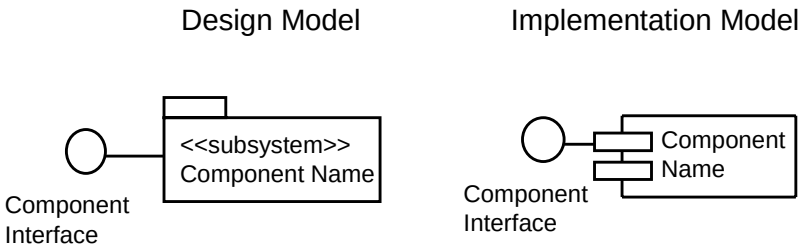
OO Principles: Encapsulation and Modularity

(stay tuned for realization relationship)

Source: Rational

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



OO Principles: Encapsulation and Modularity

Source: Rational

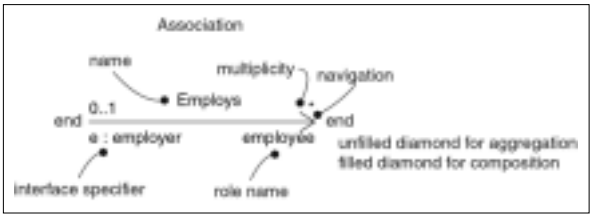
Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Source: Rational

Relationships

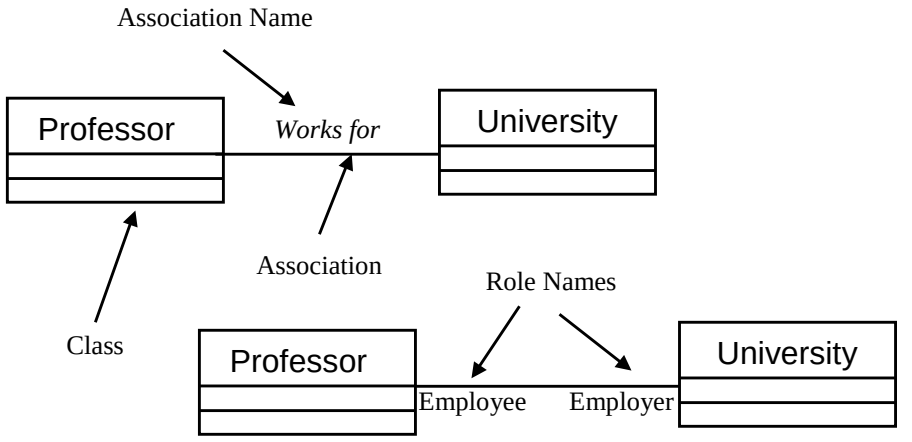
- Association
- Whole-Part
  - Aggregation
  - Composition
- Generalization
- Realization



Source: Rational

Relationships: Association

- Models a semantic connection among classes



Source: Rational

Association: Multiplicity and Navigation

- Multiplicity defines how many objects participate in a relationships
  - The number of instances of one class related to ONE instance of the other class
  - Specified for each end of the association
- Associations and aggregations are bi-directional by default, but it is often desirable to restrict navigation to one direction
  - If navigation is restricted, an arrowhead is added to indicate the direction of the navigation

Source: Rational

Association: Multiplicity

- Unspecified
- Exactly one
- Zero or more (many, unlimited)
- One or more
- Zero or one
- Specified range
- Multiple, disjoint ranges

1

0..*

*

1..*

0..1

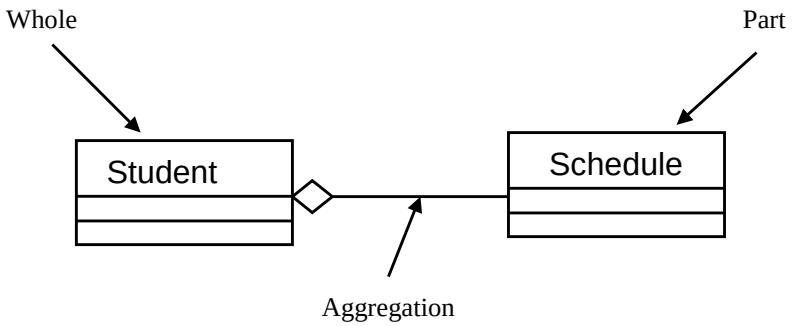
2..4

2, 4..6

Source: Rational

Relationships: Aggregation

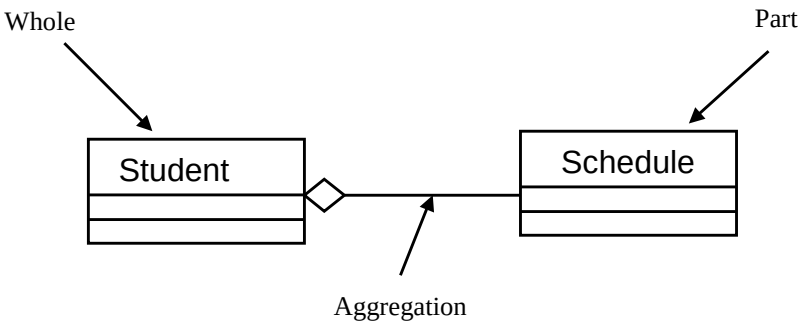
- A special form of association that models a whole-part relationship between an aggregate (the whole) and its parts



Source: Rational

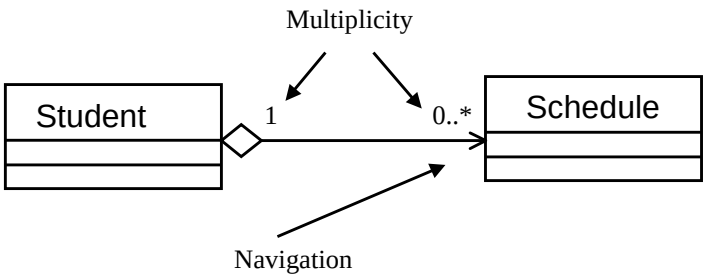
Relationships: Composition

- A form of aggregation with strong ownership and coincident lifetimes
  - The parts cannot survive the whole/aggregate



Source: Rational

Example: Multiplicity and Navigation



Source: Rational

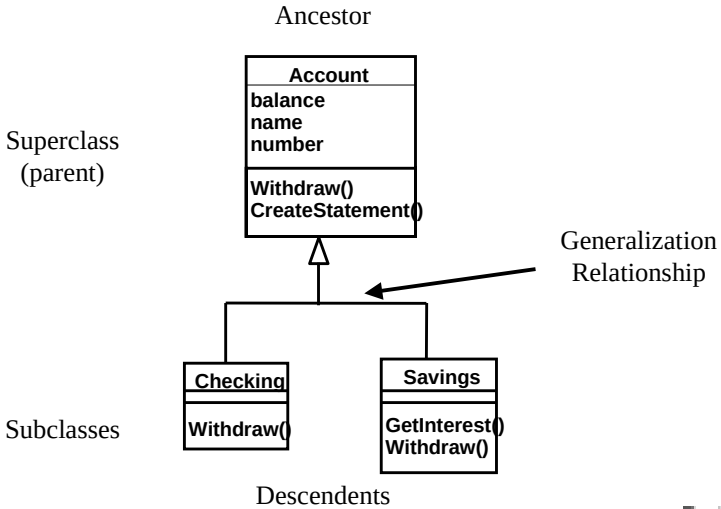
Relationships: Generalization

- 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
- Generalization is an “is-a-kind of” relationship

Source: Rational

Example: Single Inheritance

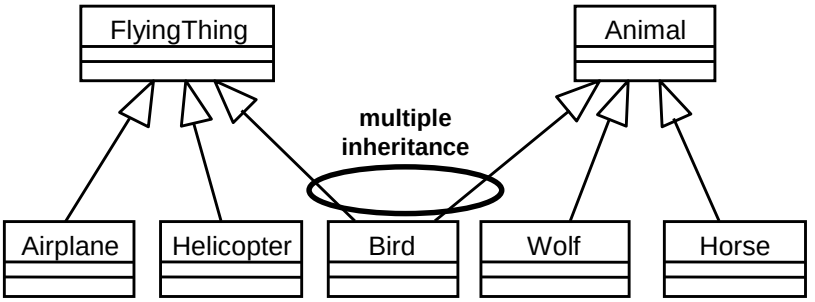
- One class inherits from another



Source: Rational

Example: Multiple Inheritance

- A class can inherit from several other classes



Use multiple inheritance only when needed, and always with caution !

Source: Rational

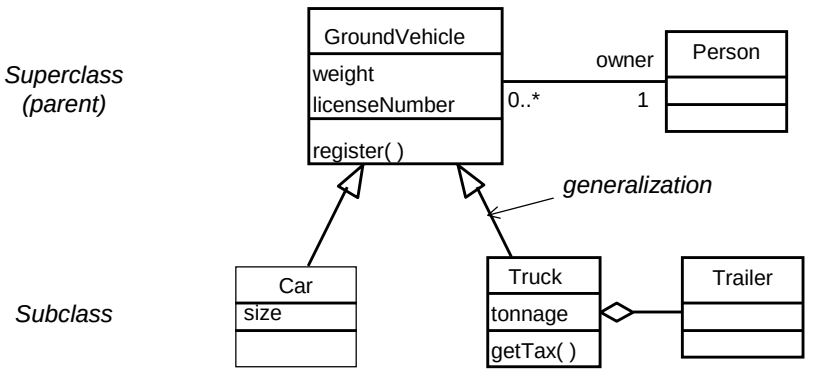
What Gets Inherited?

- 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

Source: Rational

Example: What Gets Inherited



Source: Rational

Introduction to Object Orientation Topics

- Basic Principles of Object Orientation
- Basic Concepts of Object Orientation
- Strengths of Object Orientation

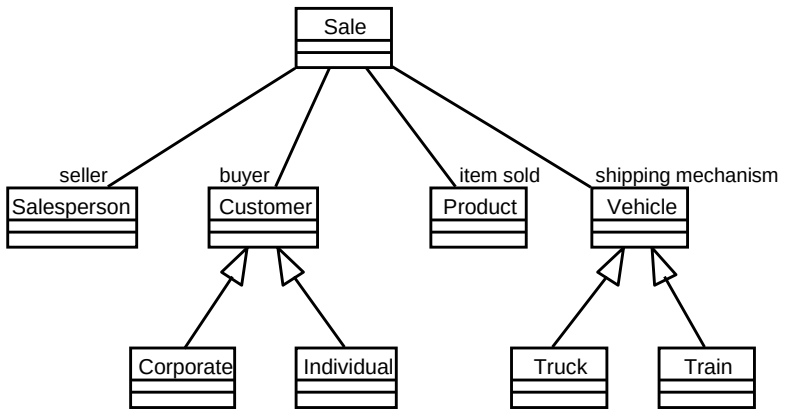
Source: Rational

Strengths of Object Orientation

- A single paradigm
- Facilitates architectural and code reuse
- Models more closely reflect the real world
  - More accurately describe corporate data and processes
  - Decomposed based on natural partitioning
  - Easier to understand and maintain
- Stability
  - A small change in requirements does not mean massive changes in the system under development

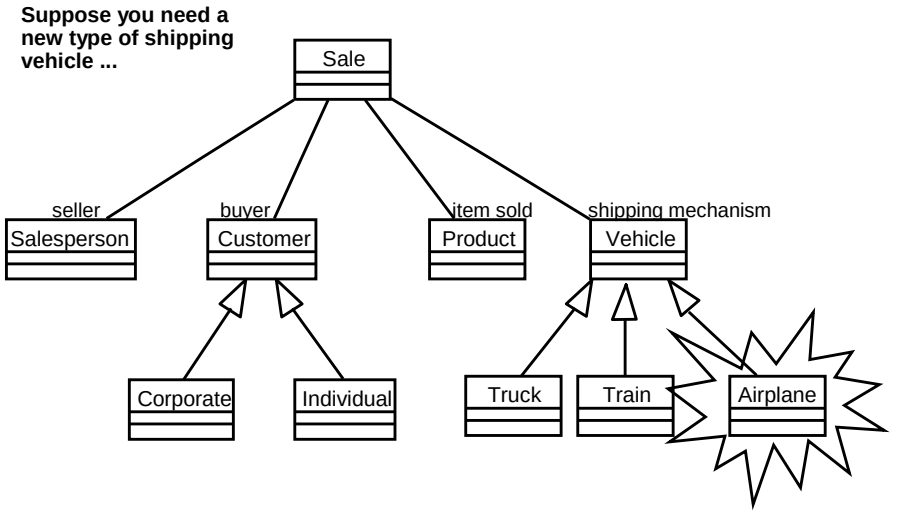
Source: Rational

Class Diagram for the Sales Example



Source: Rational

Effect of Requirements Change



Source: Rational

Let us practice the OOAD using a real case study !

☞ Development of Enterprise Project Management System

Source: Rational