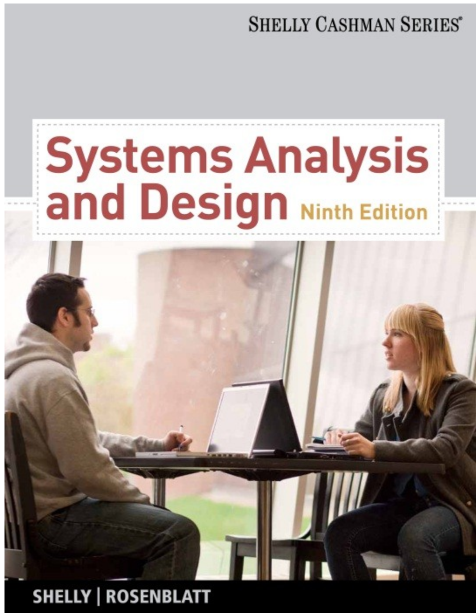
Systems Analysis and Design 9th Edition

Chapter 6

Object Modeling



Chapter Objectives

- Explain how object-oriented analysis can be used to describe an information system
- Define object modeling terms and concepts, including objects, attributes, methods, messages, classes, and instances
- Explain relationships among objects and the concept of inheritance
- Draw an object relationship diagram

Chapter Objectives

- Describe Unified Modeling Language (UML) tools and techniques, including use cases, use case diagrams, class diagrams, sequence diagrams, state transition diagrams, and activity diagrams
- Explain the advantages of using CASE tools in developing the object model
- Explain how to organize an object model

Introduction

- You learn about object-oriented analysis, which is another way to view and model system requirements
- You use object-oriented methods to document, analyze, and model the information system

Overview of Object-Oriented Analysis

- Object-oriented (O-O) analysis
- Object
- Object-oriented analysis is a popular approach that sees a system from the **viewpoint of the objects themselves** as they function and interact
- Object model

Example

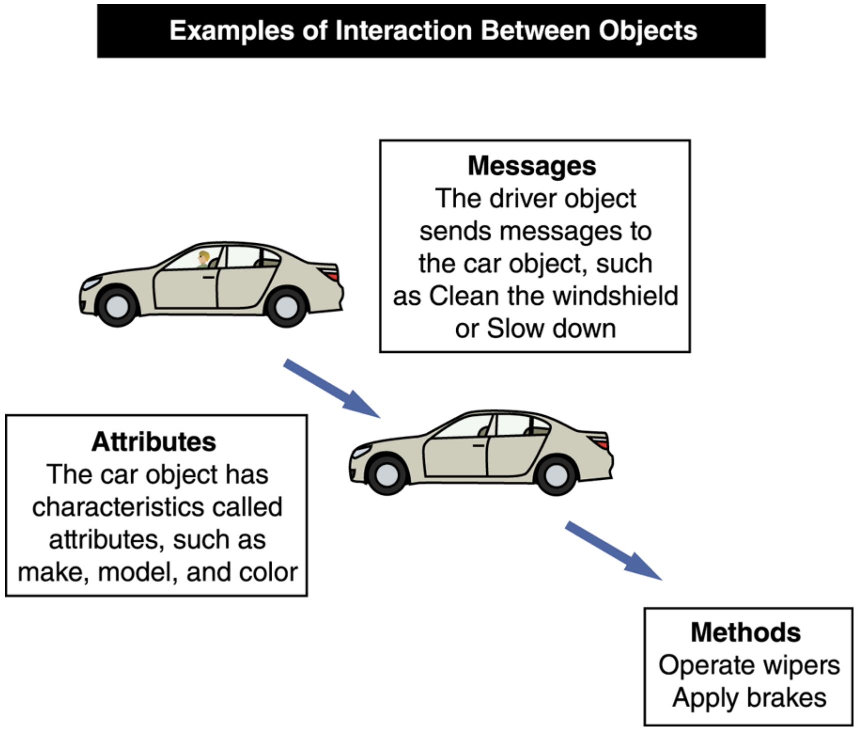
- In a **library management system**, OOA would identify:
  - **objects** such as Book, Member, and Librarian, and
  - **analyze their properties** (e.g., title, author, memberID) and
  - **interactions** (e.g., borrowing, returning)

Overview of Object-Oriented Analysis (OOA)

- **Object-Oriented Analysis (OOA)** is a method of analyzing a problem **by looking at the system as a collection of interacting objects**.
- Each object represents a real-world entity or concept and is **characterized by its attributes and behaviors**.
- The purpose of OOA is to better understand a system by **breaking it down into manageable pieces that mimic real-world entities and their interactions**

Overview of Object-Oriented Analysis

- Object-Oriented Terms and Concepts
  - Unified Modeling Language (UML)
  - Attributes
  - Methods
  - Message
  - Class
  - Instance


Examples of Interaction Between Objects
Messages
The driver object sends messages to the car object, such as Clean the windshield or Slow down
Attributes
The car object has characteristics called attributes, such as make, model, and color
Methods
Operate wipers
Apply brakes

Attributes in OOA

- **Attributes** are **properties or characteristics** that describe an object and define its state.
- If **objects** are analogous to **nouns** (e.g., "Car," "Employee," "Book"), then **attributes** are like **adjectives** that describe these nouns (e.g., color, age, title). Attributes give more information about an object's current state and can vary depending on the object.
- **Examples of Objects and Attributes: Car**

  Attributes: color (e.g., red), make (e.g., Toyota), model (e.g., Corolla), year (e.g., 2022), engine type (e.g., hybrid).

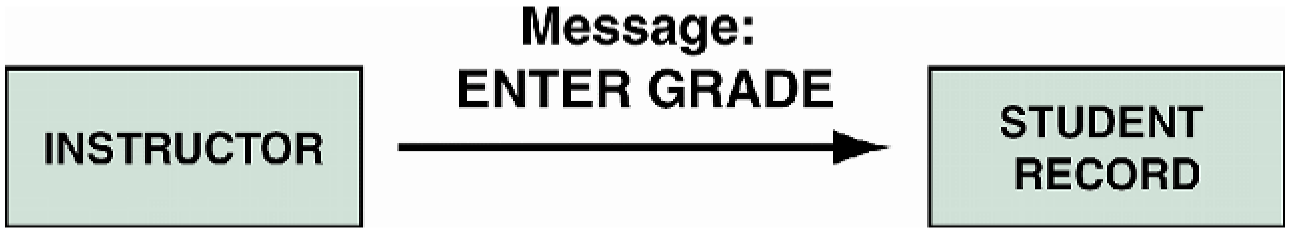
Messages

- Messages are the **means by which objects communicate and interact with each other**.
- An object sends a message to another object **to invoke a method** or perform an action.
- This helps achieve interaction and coordination among various components of the system.

- **Example:** In a Car and Driver objects, the Driver object may send a startEngine() message to the Car object. The Car object then processes this message and executes the startEngine method

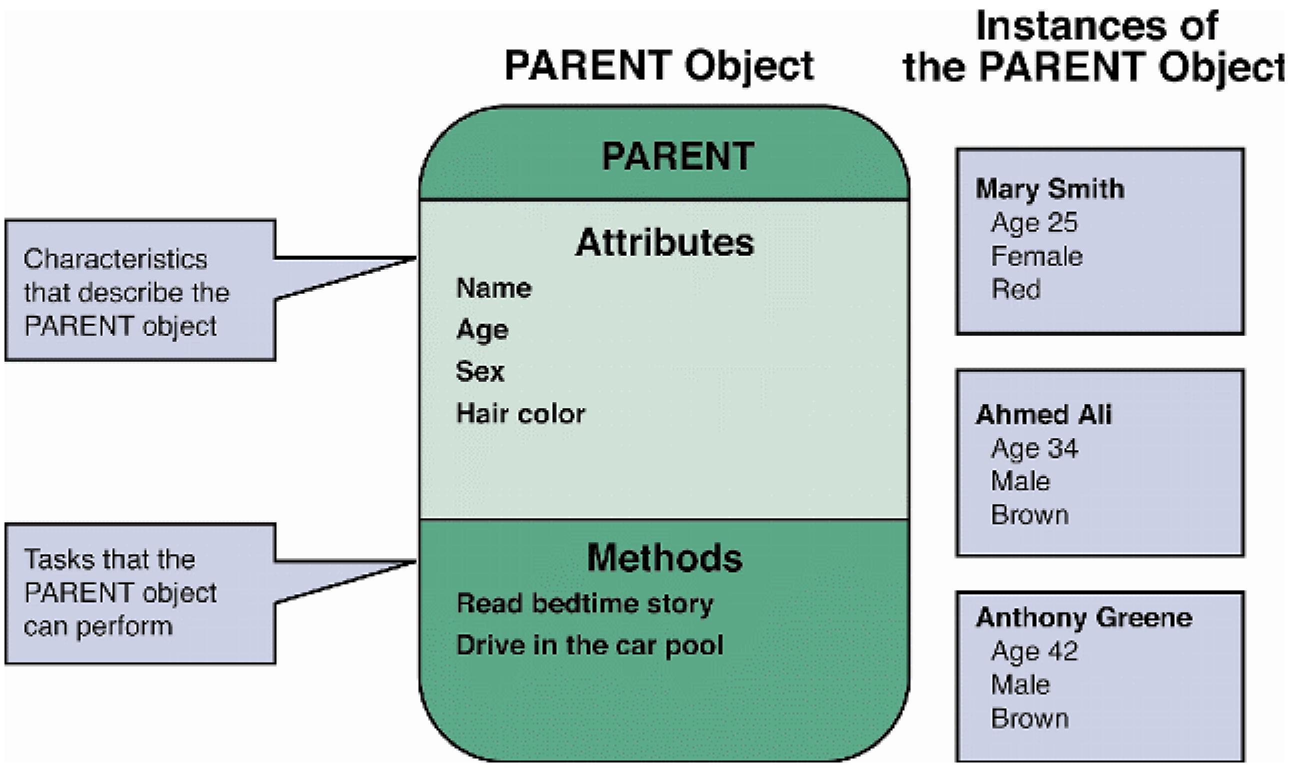
Overview of Object-Oriented Analysis

- Messages
  - A major advantage of O-O designs is that systems analysts can save time and avoid errors by using modular objects, and programmers can translate the designs into code, working with reusable program modules that have been tested and verified



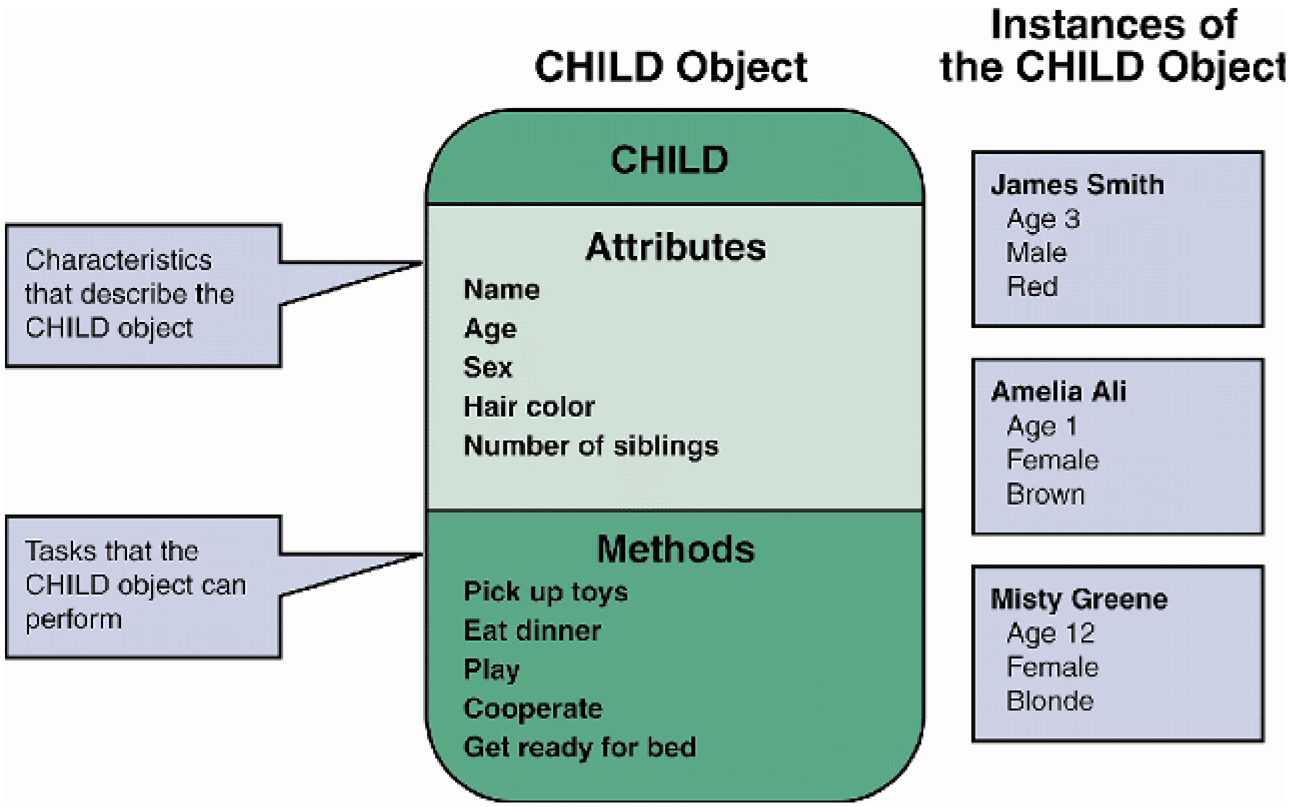
Overview of Object-Oriented Analysis

- Objects



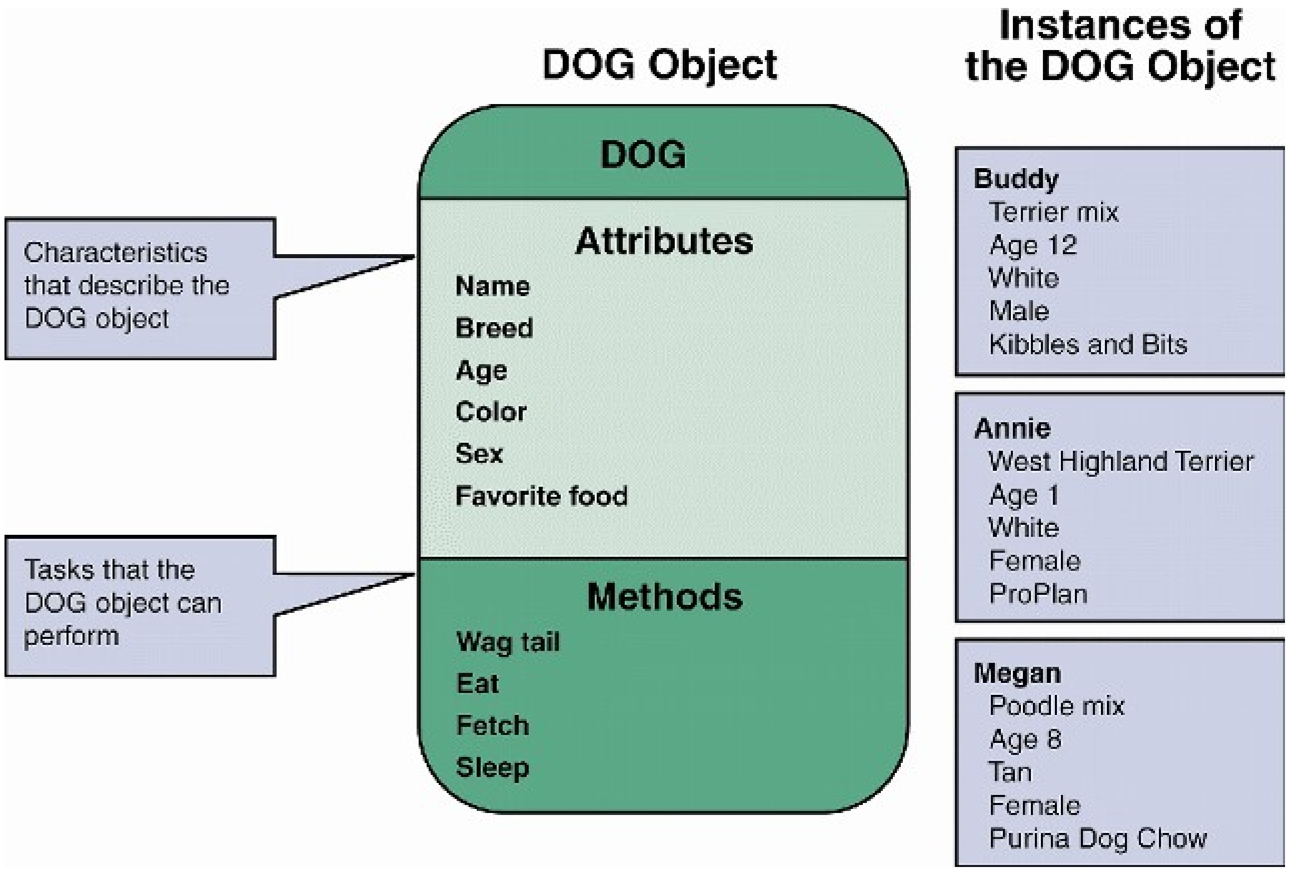
Overview of Object-Oriented Analysis

- Objects



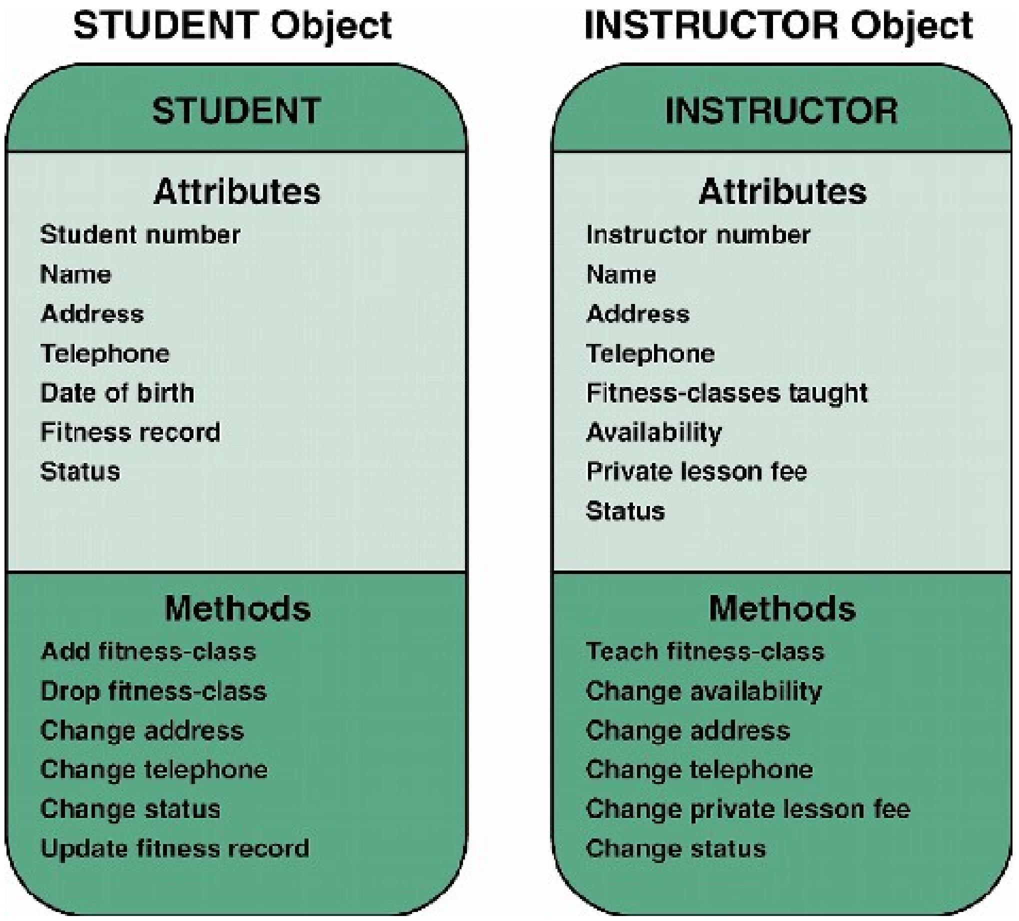
Overview of Object-Oriented Analysis

- Objects



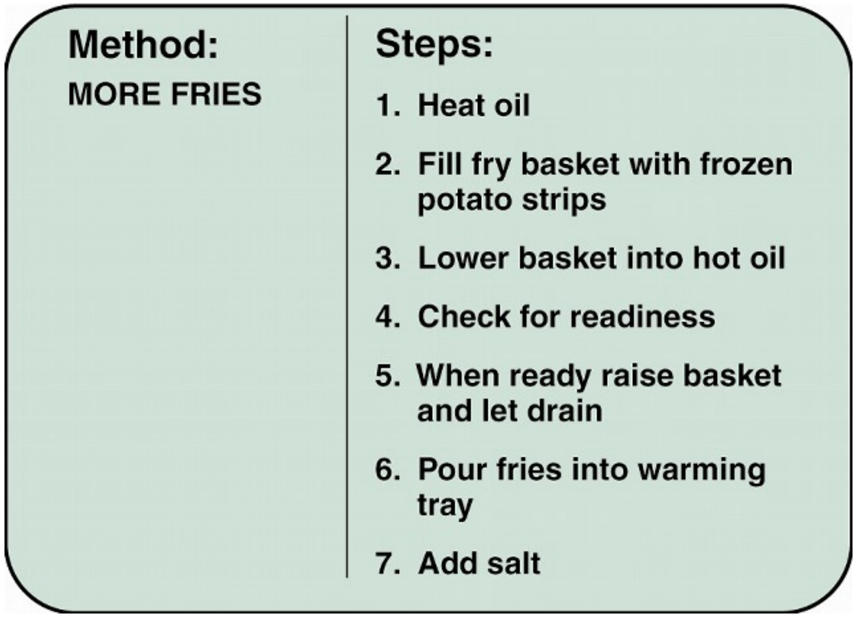
Overview of Object-Oriented Analysis

- Objects



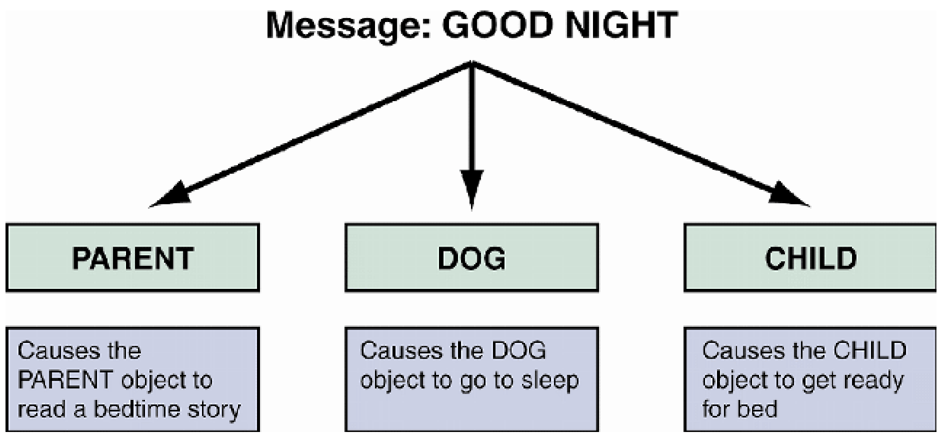
Overview of Object-Oriented Analysis

- Methods
  - A method defines specific tasks that an object can perform
  - Just as objects are similar to nouns and attributes are similar to adjectives, **methods** resemble **verbs** that describe what and how an object does something



Method:	**Steps:**
MORE FRIES	1. Heat oil 2. Fill fry basket with frozen potato strips 3. Lower basket into hot oil 4. Check for readiness 5. When ready raise basket and let drain 6. Pour fries into warming tray 7. Add salt

Overview of Object-Oriented Analysis


Message: GOOD NIGHT
PARENT
DOG
CHILD
Causes the PARENT object to read a bedtime story
Causes the DOG object to go to sleep
Causes the CHILD object to get ready for bed

- Messages
  - Polymorphism
  - Black box
  - Encapsulation

Polymorphism

Polymorphism is the ability of **objects of different types** to be treated as **objects** of a **common super type**. It allows a single method or function to operate in different ways based on the object it is applied to.

This leads to more flexible and **reusable code** by allowing methods to be overridden or used across different classes

Example: Consider a base **class Shape** with a method **draw()**. Different derived classes like **Circle**, **Rectangle**, and Triangle override the draw() method to implement their specific drawing logic. **A piece of code that calls draw() on a Shape object can work with any subclass without knowing its exact type**

Black Box

Black box refers to the concept of **hiding the internal implementation details of an object and only exposing its behavior or interface**.

This means that the user of an object **only needs to know how to interact with it** (i.e., which methods are available) without needing to understand how those methods are implemented

Example: A Car class may have methods like startEngine() and accelerate().

The user of the Car class **doesn't need to know how the engine starts internally**; they only need to call startEngine()

Encapsulation

Encapsulation is the practice of bundling the data (attributes) and methods (functions) that operate on that data **within a single unit, typically a class**.

It restricts direct access to some of an object’s components, which means the internal representation of the object is hidden from the outside.

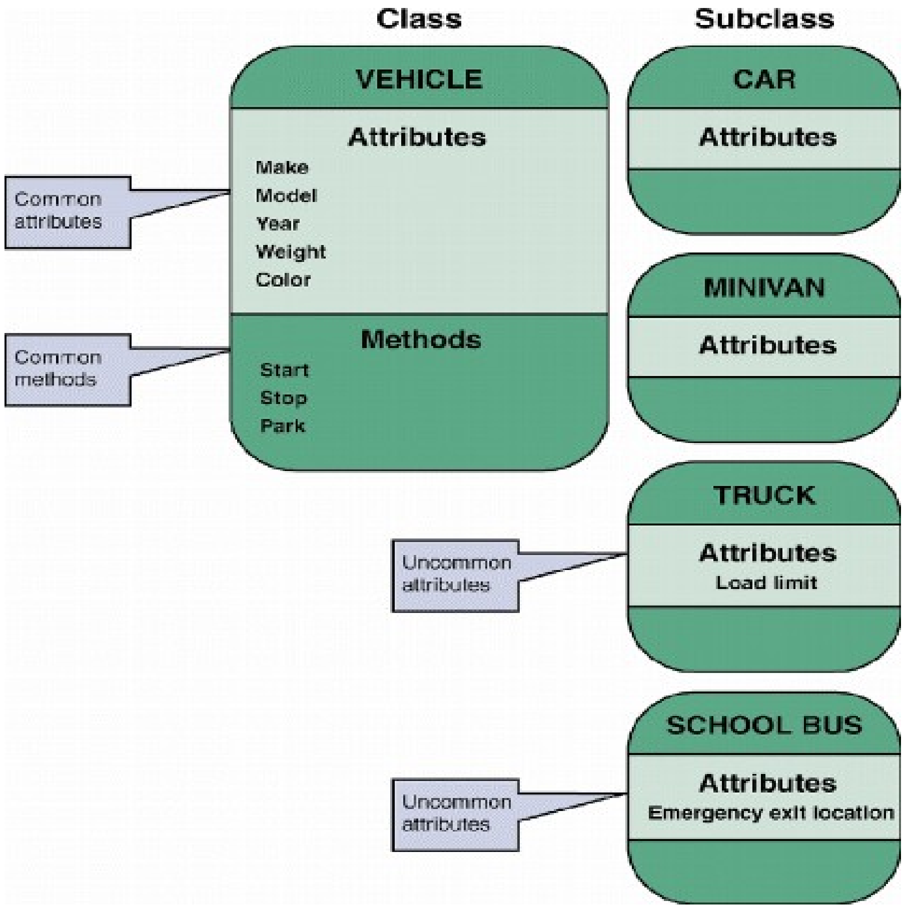
Example: A BankAccount class might have private data members such as balance and methods deposit() and withdraw(). **External code cannot directly access or modify balance**; it must use the provided methods, which can include checks and safeguards

Overview of Object-Oriented Analysis

- Classes
  - An object belongs to a group or category called a class
  - All objects within a class share common attributes and methods
  - Subclasses
  - Superclass

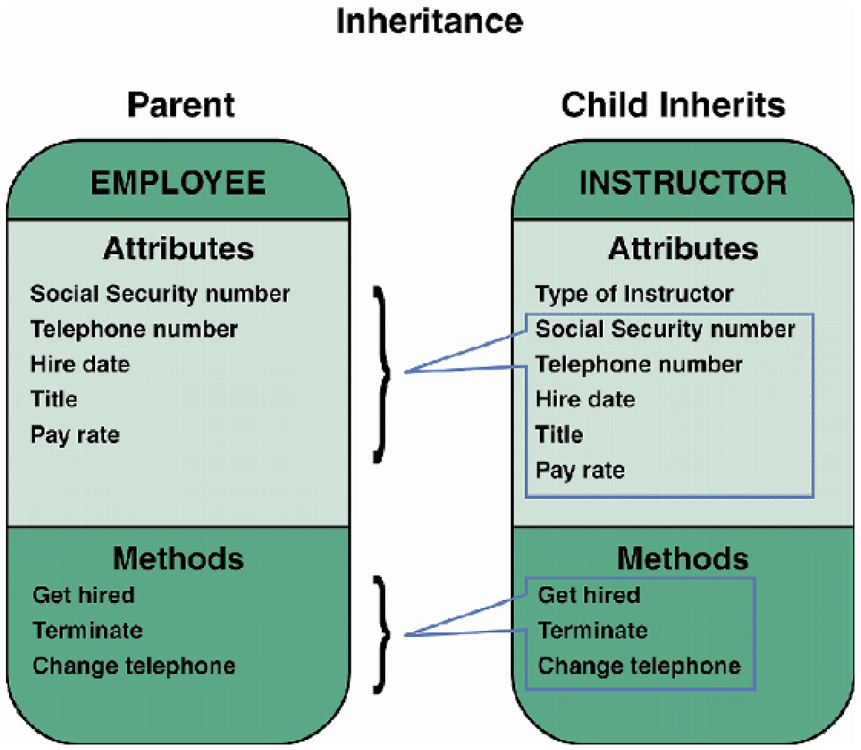
Overview of Object-Oriented Analysis

- Classes



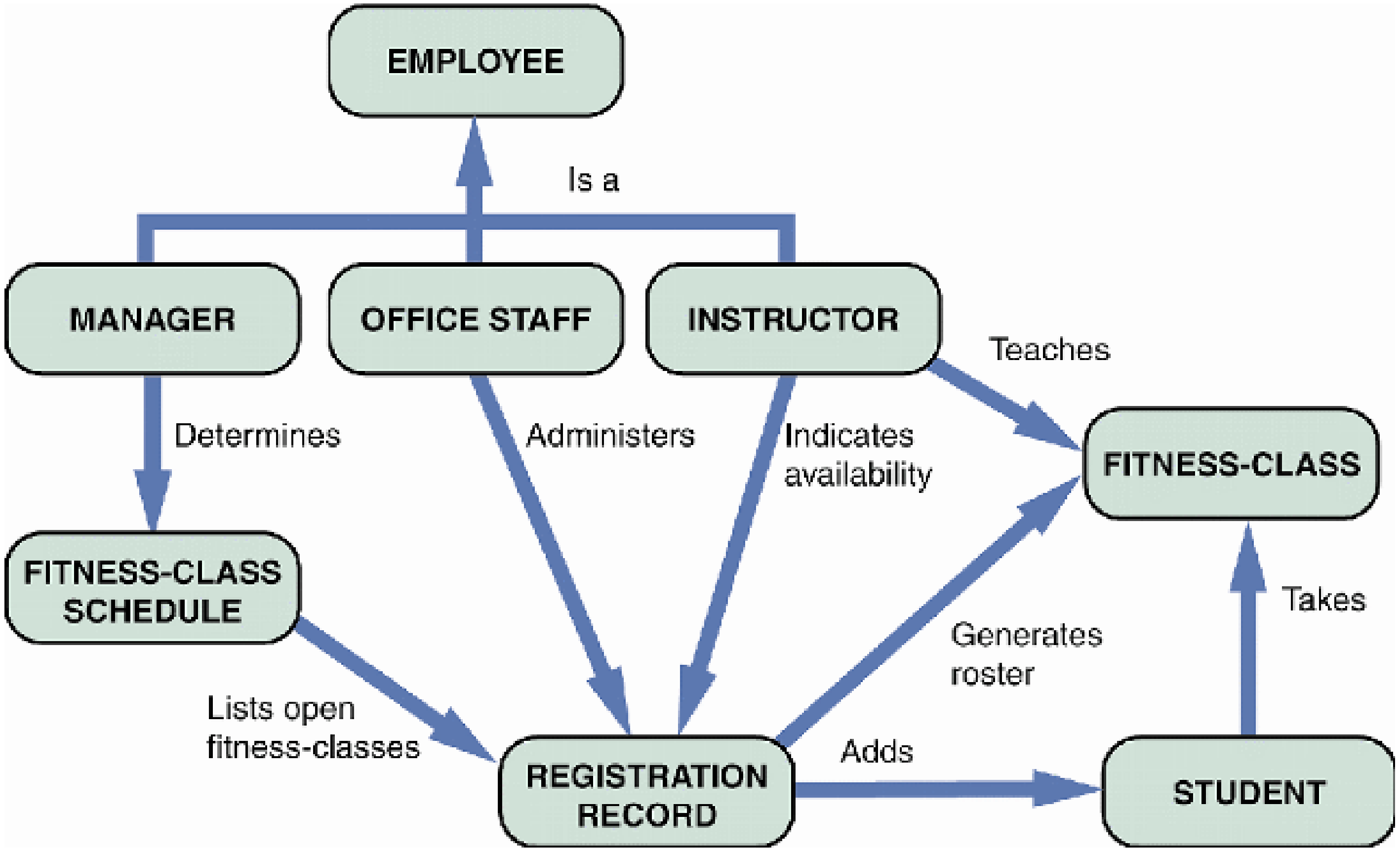
Relationships Among Objects and Classes

- Inheritance
- Child
- Parent



Relationships Among Objects and Classes

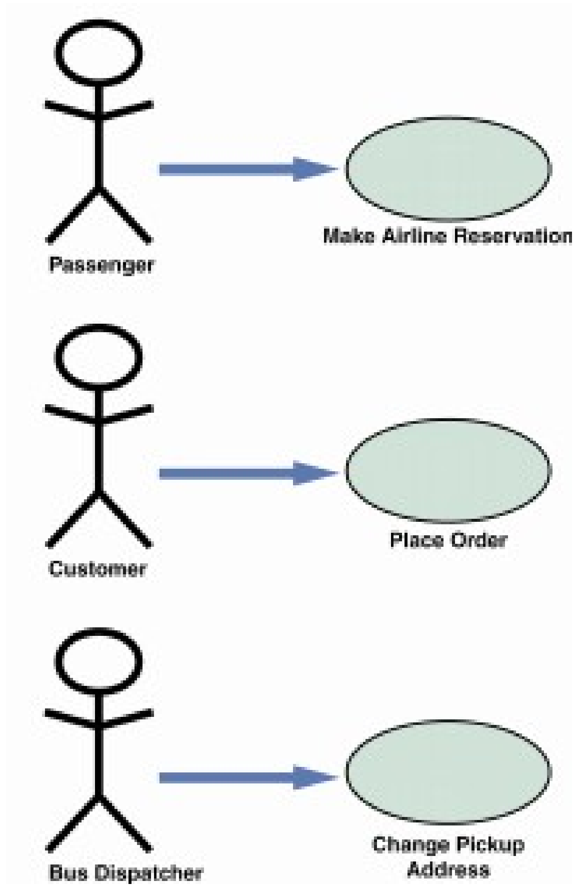
- Object Relationship Diagram



Object Modeling with the Unified Modeling Language

- The UML uses a set of symbols to represent graphically the various components and relationships within a system
- It mainly is used to support object-oriented systems analysis and to develop object models

Object Modeling with the Unified Modeling Language



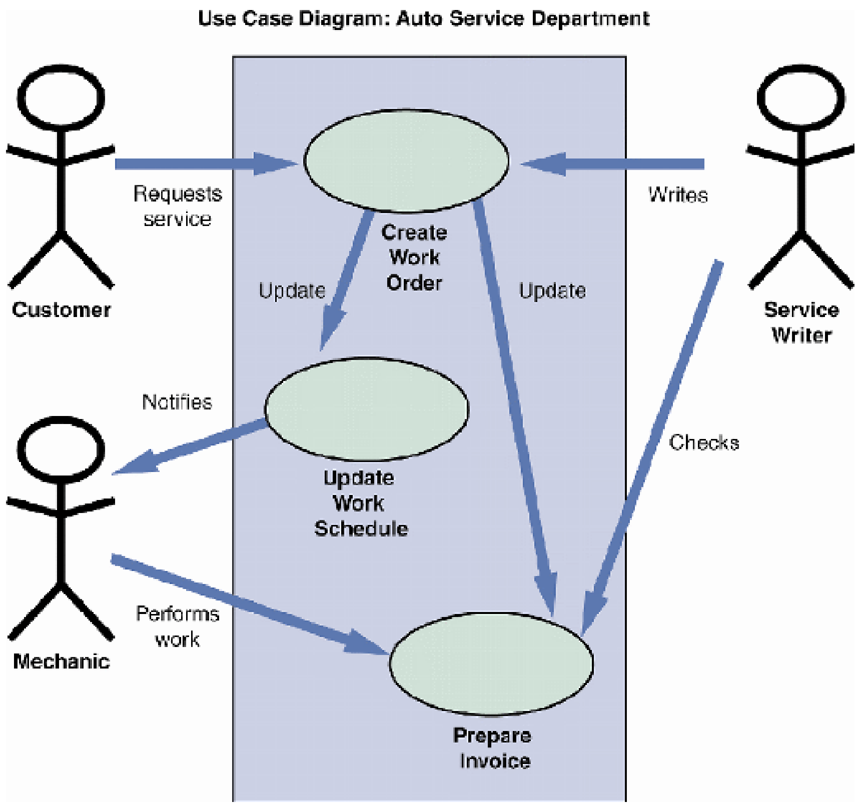
- Use Case Modeling
  - Actor
  - Symbol for a use case is an oval with a label that describes the action or event
  - Use cases also can interact with other use cases

Object Modeling with the Unified Modeling Language

- Use Case Modeling
  - When the outcome of one use case is incorporated by another use case, we say that the second case uses the first case
  - Use case description
  - When you identify use cases, try to group all the related transactions into a single use case

Object Modeling with the Unified Modeling Language

- Use Case Diagrams
  - Use case diagram
  - System boundary
  - After you identify the system boundary, you place the use cases on the diagram, add the actors, and show the relationships

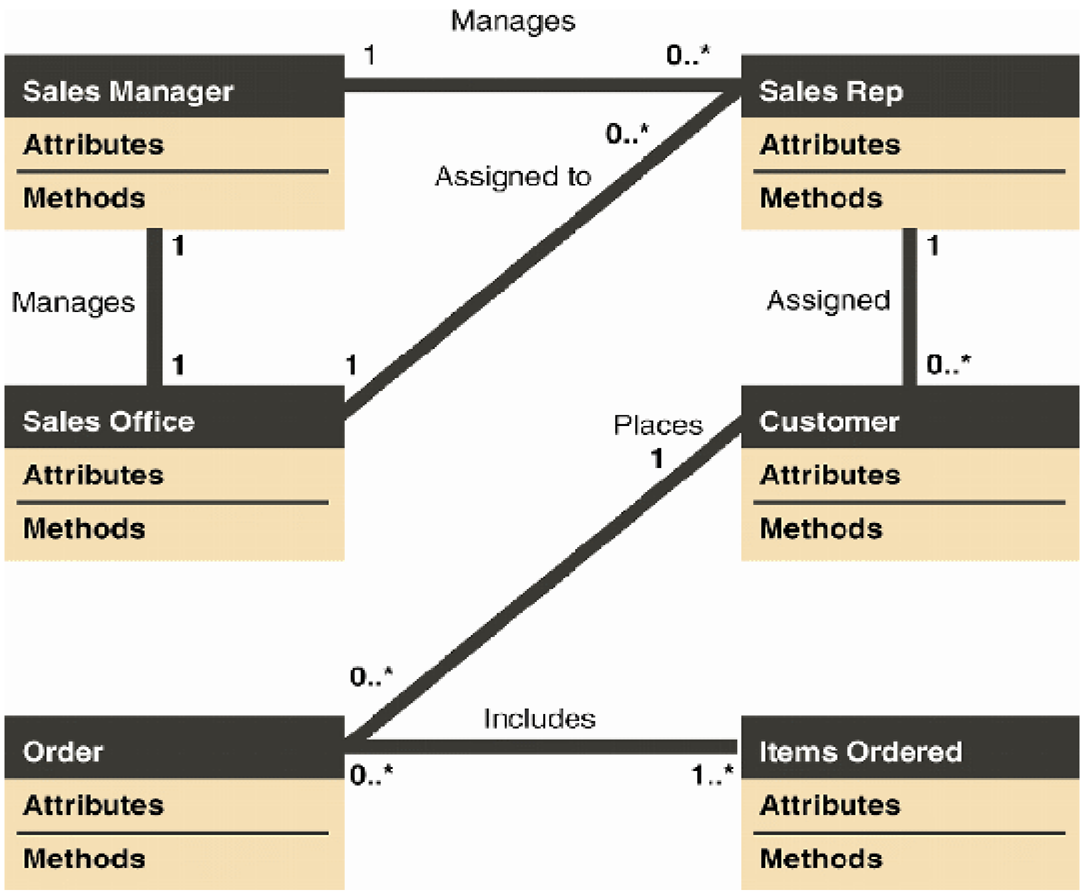


Object Modeling with the Unified Modeling Language

- Class Diagrams
  - Class Diagram
  - Evolves into a physical model and finally becomes a functioning information system
  - Each class appears as a rectangle, with the class name at the top, followed by the class’s attributes and methods
  - Cardinality

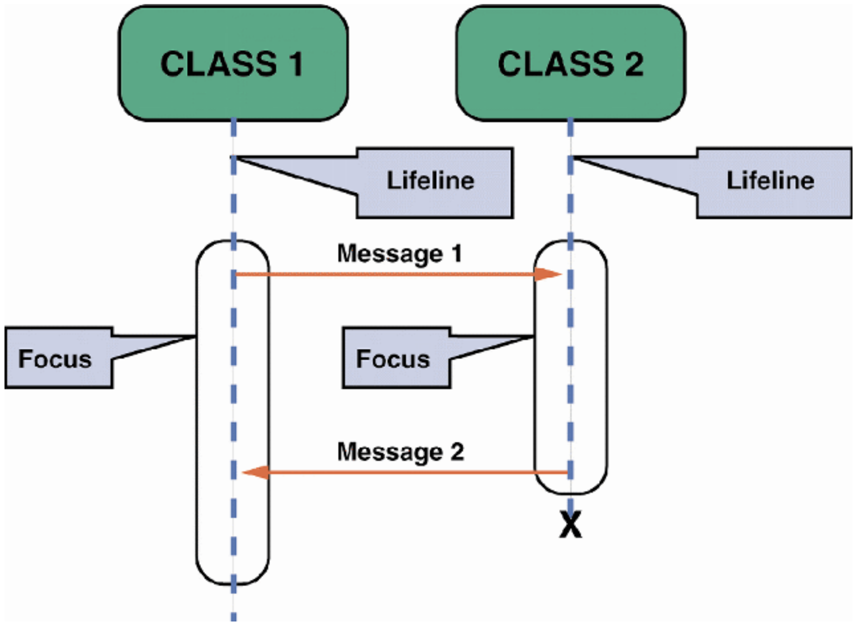
Object Modeling with the Unified Modeling Language

- Class Diagrams



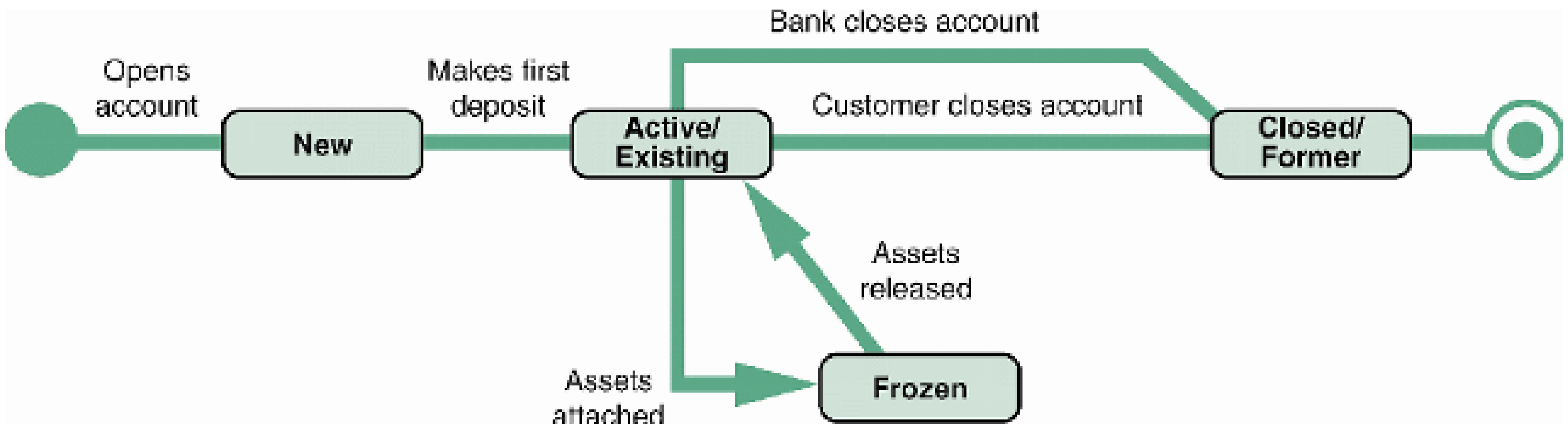
Object Modeling with the Unified Modeling Language

- Sequence Diagrams
  - Sequence diagram
  - Include symbols that represent
    - Classes
    - Lifelines
    - Messages
    - Focuses



Object Modeling with the Unified Modeling Language

- State Transition Diagrams

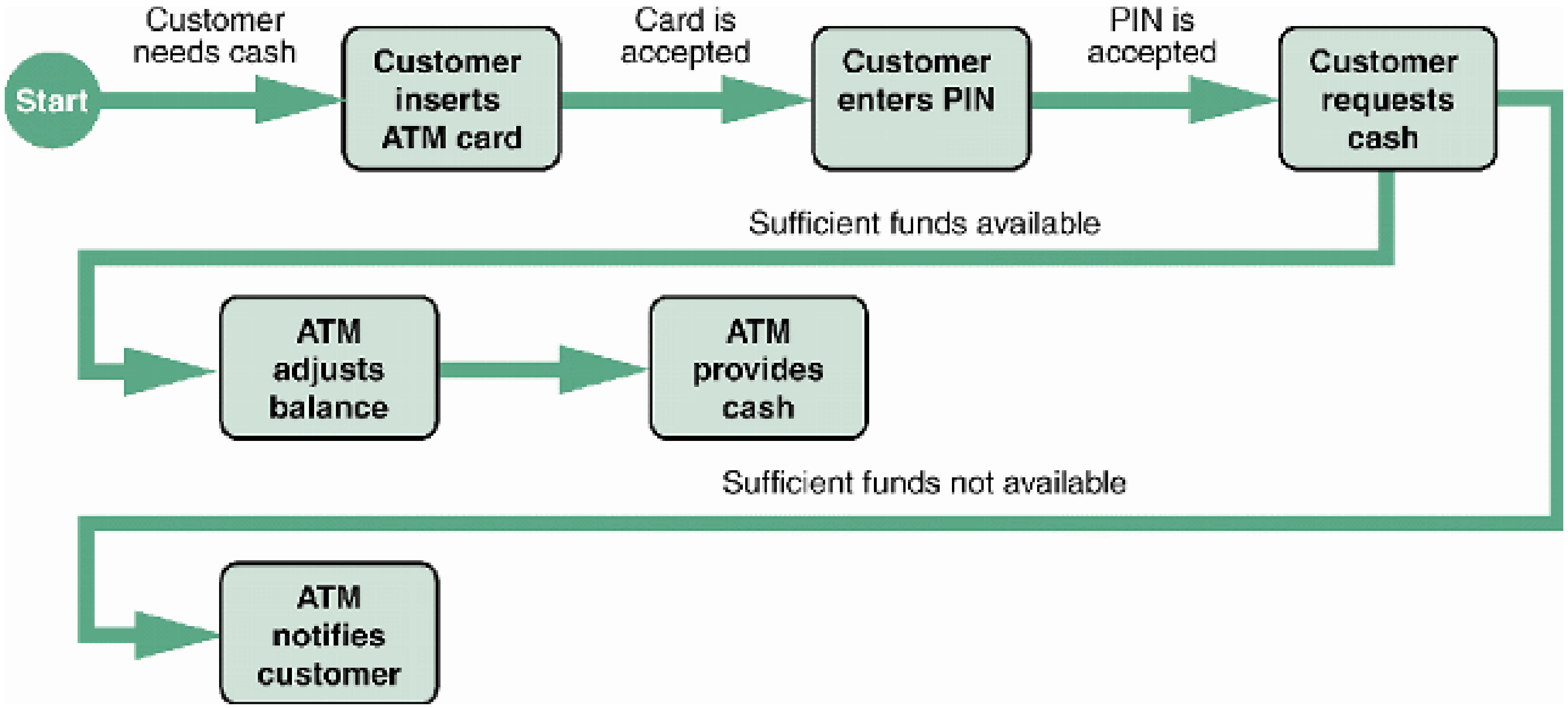


Object Modeling with the Unified Modeling Language

- State Transition Diagrams
  - The small circle to the left is the initial state, or the point where the object first interacts with the system
  - Reading from left to right, the lines show direction and describe the action or event that causes a transition from one state to another
  - The circle at the right with a hollow border is the final state

Object Modeling with the Unified Modeling Language

- Activity Diagrams



Object Modeling with the Unified Modeling Language

- Activity Diagrams
  - Sequence diagrams, state transition diagrams, and activity diagrams are dynamic modeling tools that can help a systems analyst understand how objects behave and interact with the system

Object Modeling with the Unified Modeling Language

- CASE Tools
  - Object modeling requires many types of diagrams to represent the proposed system
  - Creating the diagrams by hand is time-consuming and tedious, so systems analysts rely on CASE tools to speed up the process and provide an overall framework for documenting the system components

Organizing the Object Model

- You should develop an object relationship diagram that provides an overview of the system
- You should organize your use cases and use case diagrams so they can be linked to the appropriate class, state transition, sequence, and activity diagrams
- It is much easier to repair a diagram now than to change the software later

Chapter Summary

- This chapter introduces object modeling, which is a popular technique that describes a system in terms of objects
- The Unified Modeling Language (UML) is a widely used method of visualizing and documenting an information system
- At the end of the object modeling process, you organize your use cases and use case diagrams and create class, sequence, state transition, and activity diagrams

Chapter Summary

- Chapter 6 complete