6.1 Object-Oriented Analysis

* Methodology is popular
  + Integrates easily with object-oriented programming languages such as C++, Java, and Python
  + Modular, reusable, and easy to maintain
* End product of O-O analysis is an object model
  + Represents the information system in terms of object and O-O concepts

6.2 Objects

* Person, place, event, or transaction that is significant to the information system

6.3 Attributes

* Describe the characteristics of an object
  + Attributes of an object are defined during the system development process
* Objects possess a state
  + Describes the object’s current status

6.4 Methods

* Tasks or functions that the object performs when it receives a message, or command

6.5 Messages

* Command that tells an object to perform a certain method
  + Polymorphism: message gives different meanings to different objects
* Message to the object triggers changes within the object without specifying how the changes must be carried out
  + An object can be viewed as black box
    - Encapsulation: idea that all data and methods are self-contained

6.6 Classes

* An object belongs to a group or category called a class
  + All objects within a class share common attributes and methods
    - Subclasses: categories within a class
    - Super-class: class belonging to a general category

6.7 Relationships Among Objects and Classes

* Relationships
  + Enable objects to communicate and interact as they perform business functions and transactions
  + Describe what objects need to know about each other
* Inheritance
  + Strongest relationship
  + Enables an object to derive one or more of its attributes from another object

6.8 The Unified Modeling Language (UML)

* Uses a set of symbols to represent graphically the various components and relationships within a system
  + Use case modeling
    - Use case: represents steps in a specific business function or process
    - An external entity, called an actor, initiates a use case by requesting the system to perform a function or process
  + Use case description
    - Documents the name of the use case, the actor, a description of the use case
    - Provides a step-by-step list of the tasks and other key descriptions and assumptions
  + Use case diagrams
    - Visual summary of several related use cases within a system or subsystem
    - The first step is to identify the system boundary which is represented by a rectangle
      * System boundary: shows what is included in the system (inside the rectangle) and what is not included in the system (outside the rectangle)
  + Class diagrams
    - Show the object classes and relationships involved in a use case
      * Each class appears as a rectangle, with the class name at the top, followed by the class’s attributes and methods
      * Lines show relationships between classes and have labels identifying the action that relates the two classes
    - Includes a concept called cardinality
      * Describes how instances of one class relate to instances of another class
  + Sequence diagrams
    - Dynamic model of a use case, showing the interaction among classes during a specified time period
    - Graphically document the use case by showing classes, messages, and timing messages
    - Include symbols that represent classes, lifelines, messages, and focuses
  + Important terms
    - Classes: send or receive messages
    - Lifelines: represent the time during which the object above it is able to interact with the other objects in the use case
    - Messages: include additional information about the contents
    - Focuses: indicate when an object sends or receives message
  + State transition diagrams
    - Show how an object changes from one state to another, depending on events that affect the object
    - All possible setates must be documented in the state transition diagram
    - States appear as rounded rectangles with the state names inside
  + Activity diagrams
    - Show actions and events as they occur
    - Show the order in which the actions take place and identify the outcomes
  + Business process modeling (BPM)
    - Represents the people, events, and interaction in a system
    - Can be used anytime during the systems development process
    - Compatible with object modeling

6.9 Tools

* Object modeling requires many types of diagrams to represent the proposed system
  + Systems analysts rely on tools to speed up the process and provide an overall framework for documenting
    - CASE tools
    - Systems modelling tools