Object Modeling

Peter Lo

Object Oriented Terms & Concepts

- An **Object** represents a real person, place, event, or transaction.
- A **Class** is a group of similar objects.
- An **Instance** is a specific member of a class.

Objects

- An **Object** is an encapsulation of state (attributes, data variables) and procedures (operations, methods, member functions) that operate on the state; an object has a unique identity and an interface.

Attributes

- Attributes describe the characteristics of an object.
- Objects can have a specific attribute called a state.
- The state of an object describes the object's current status.

STUDENT Object

State	Status
Future	Registered, but has not started to attend
Current	Registered, attending one or more fitness-classes
Past	Attended one or more fitness-classes in the past

FITNESS-CLASS Object

State	Status
Open	Fitness-class is open for enrollment
Closed	Fitness-class has reached maximum enrollment
Canceled	Fitness-class has been canceled

BANK ACCOUNT Object

State	Status
Active	Account is open and meets standards for activity
Inactive	Account is dormant for specified period of time
Closed	Account was closed
Frozen	Assets have been legally attached

Method

- A **Method** defines specific tasks that an object can perform.
- Example:
  - ***Constructor Method:*** A method that creates a new instance of an object.
  - ***Update Method:*** A method that changes existing data.
  - ***Query Method:*** Method that provides information about an object's attributes

Message

- A **Message** is a command that tells an object to perform a certain method.

Polymorphism

- The same message to two different objects can produce different results.
- The concept that a message gives different meanings to different objects is called **Polymorphism**.
- Definition of Polymorphism: ***The ability to manipulate objects of distinct classes using only knowledge of their common properties without regard for their exact class***

Encapsulation

- An object can viewed as a black box because a message to the object triggers changes within the object without specifying how the changes must be carried out.
- The black box concept is an example of **Encapsulation:** all data and methods are self-contained.
- Definition of Encapsulation: *The object encapsulates or hides both data and the logical procedures required to manipulate the data*

Classes

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

Relationships among Objects and Classes

- Relationships describe:
  - what objects need to know about each other
  - how objects respond to changes in other objects
  - the effects of membership in classes, super-classes, and sub-classes

Dependency

- Dependency occurs when one object must be informed about another.

Association

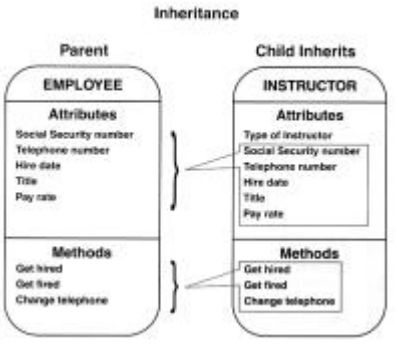
- Occurs when certain attributes of one object are determined by its interaction with another objects.

Aggregation

- Aggregation (part-of) relationship: An object can be composed of many component objects and may itself be a component object.
  - E.g., Airplane is an object composed of hundreds or thousands of component objects, such as wheels, wings, instruments, etc.

Inheritance

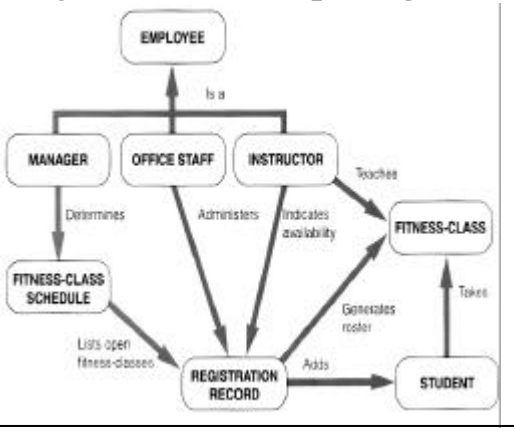
- Inheritance enables an object to derive one or more of its attributes from another object.



Object Relationship Diagram

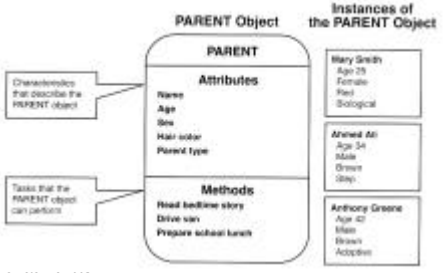
- An object relationship diagram provide an overview of the system

Object Relationship Diagram



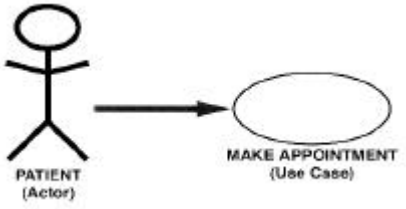
Object Modeling with UML

- The **Unified Modeling Language** (UML) represents an object as a rectangle with the object name at the top followed by the object's attributes and methods.



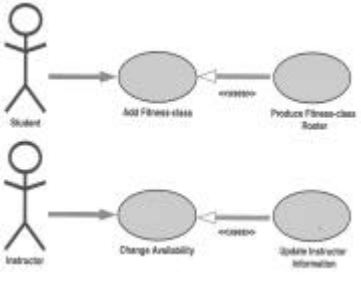
Use Case Modeling

- A **Use Case** represents the 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 Modeling

- Use cases also can interact with other use cases. 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 Modeling

- For each use case, you also develop a use case description in the form of a table.
- A use case description documents the name of the use case, the actor, a description of the use case, a step-by-step list of the tasks and actions required for successful completion, a description of alternative courses of action, pre-conditions, post-conditions, and assumptions.

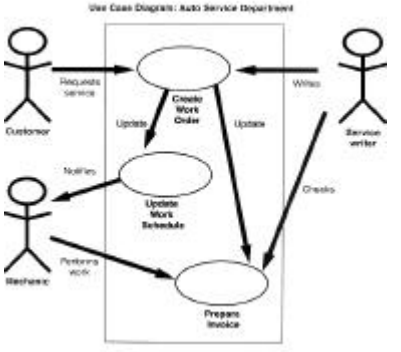
Use Case Description Example

Add New Student Use Case

Name:	Add New Student
Actor:	Student/Manager
Description:	Describes the process used to add a student to a fitness-class
Successful completion:	1. Manager checks FITNESS-CLASS SCHEDULE object for availability 2. Manager notifies student 3. Fitness-class is open and student pays fee 4. Manager registers student
Alternative:	1. Manager checks FITNESS-CLASS SCHEDULE object for availability 2. Fitness-class is full 3. Manager notifies student
Precondition:	Student requests fitness-class
Postcondition:	Student is enrolled in fitness-class and fees have been paid
Assumptions:	None

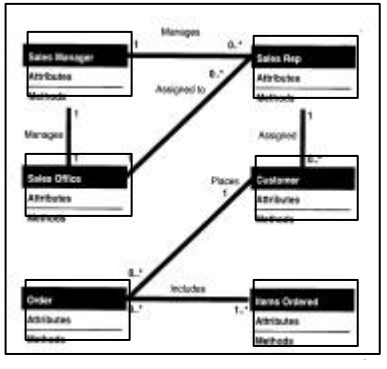
Use Case Diagrams

- A use case diagram is a visual summary of several related use cases within a system or sub-system.



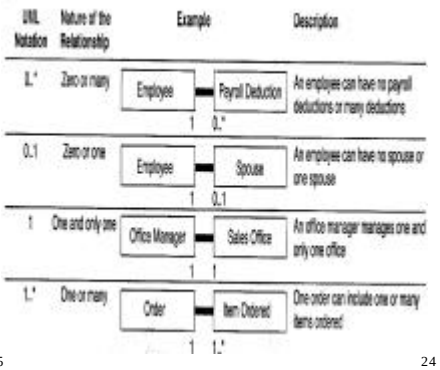
Class Diagrams

- A class diagram represents a detailed view of a single use case
- It shows the classes that participate in the use case, and documents the relationship among the classes.



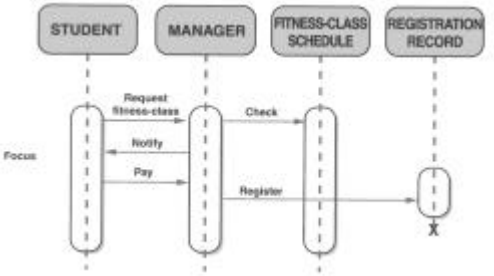
Cardinality

- Cardinality describes how instances of one class relate to instances of another class



Sequence Diagrams

- A sequence diagram is a dynamic model of a use case
- It show the interaction among classes during a specified time period.



Sequence Diagrams

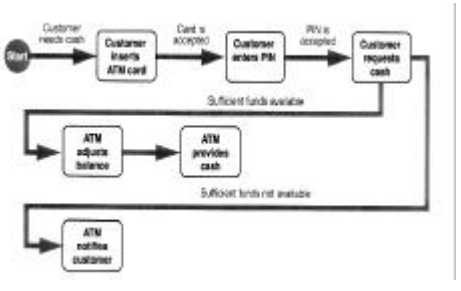
- CLASSES
  - A class is identified by a rectangle with the name inside.
- LIFELINES
  - A dashed line as shown in the Figure below identifies a lifeline.
- MESSAGES
  - A line showing direction that runs between two objects, as shown in figure below, identifies a message.
- FOCUSES
  - A focus is identified by a narrow vertical rectangle that covers the lifeline.

State Transition Diagrams

- A **State Transition Diagram** shows how an object changes from one state to another, depending on events that affect the object.

Activity Diagrams

- An activity diagram resembles a horizontal flow chart that shows the actions and events as they occur.
- It show the order in which the actions take place and identify the outcomes.



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.
- **CASE tools** speed up the development process, ensure consistency, and provide common links so that objects can be described in one part of the design and reused in other areas of the model.

Organizing the Object Model

- Organize your use cases and use case diagrams so they can be linked to the appropriate class, state transition, sequence, and activity diagrams.
- Those diagrams will form the foundation for the systems design, so accuracy is important.
- You can proceed to evaluating alternatives, creating the systems requirement document, delivering a presentation to management, and preparing for the transition to systems design.