Review of Object-oriented Concepts

Topics

- Object Technologies
- Objects
  - Object's State
  - Object's Behavior
  - Object's Identity
- Four Basic Principles of Object-orientation
  - Abstraction
  - Encapsulation
  - Modularity
  - Hierarchy

Object-oriented Software Engineering

- Object-oriented Software Engineering is the use of object technologies in building software.

Object Technology

- Pressman, 1997
  - Object technologies is often used to encompass all aspects of an object-oriented view and includes analysis, design, and testing methods; programming languages; tools; databases; and applications that are created using object-oriented approach.
- Taylor, 1997, Object Technology
  - Object technology is a set of principles guiding software construction together with languages, databases, and other tools that support those principles.

Benefits of Object Technology

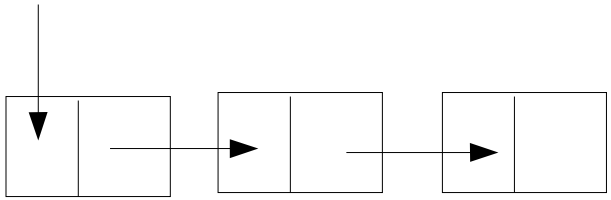
- It leads to reuse, and reuse (of program components) leads to faster software development and higher-quality programs.
- It leads to higher maintainability of software modules because its structure is inherently decoupled.
- It leads to object-oriented system that are easier to adapt and easier to scale, ie, large systems are created by assembling reusable subsystems.

Object

- It is a representation of an entity either physical, conceptual, or software.
- It allows software developers to represent real-world concepts in their software design.

Airplane

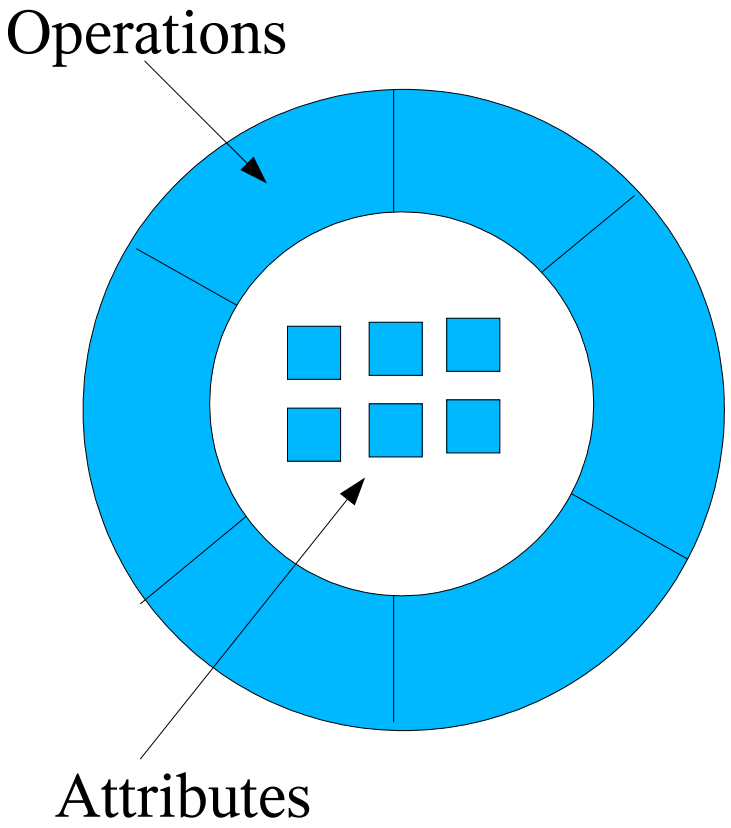
Chemical Process



Linked List

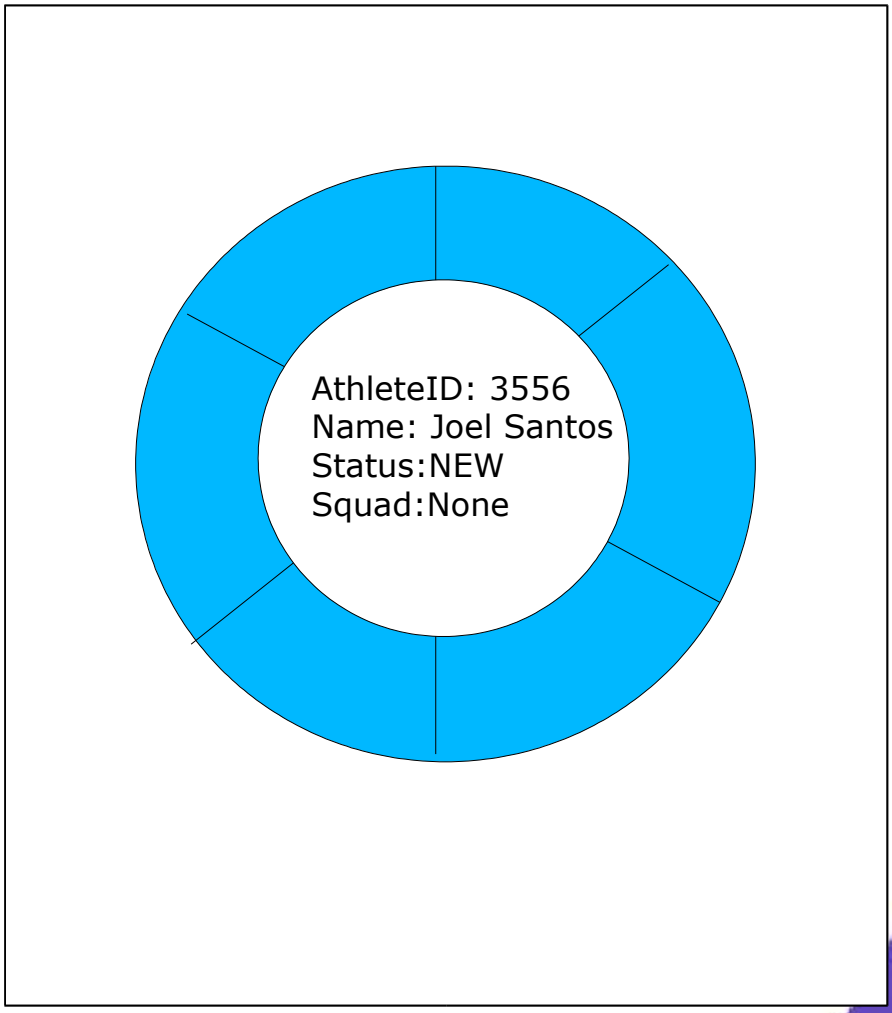
Object

- It is an entity with a well-defined boundary and identity that encapsulates state and behavior.



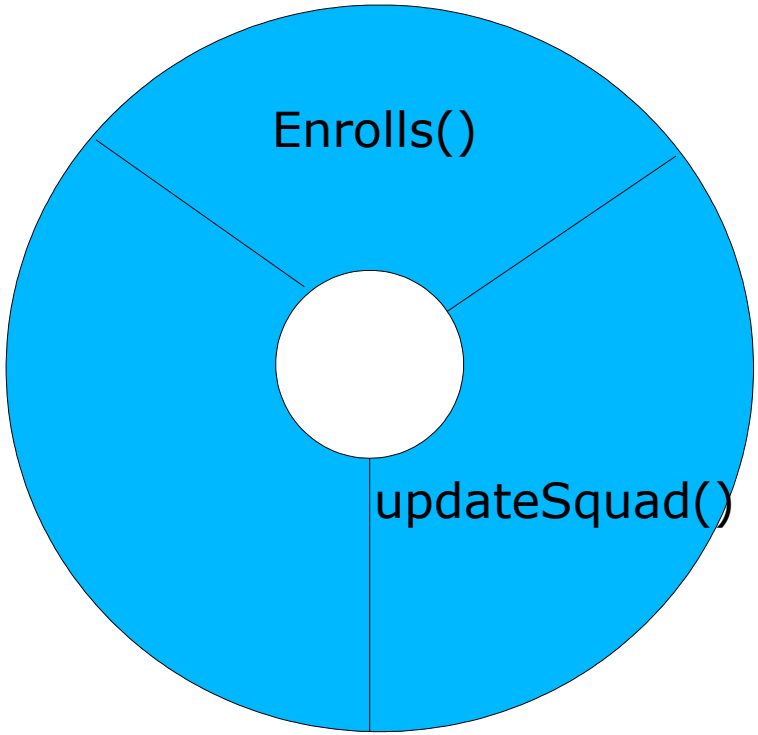
Object's State

- It is one of the possible conditions that an object may exists in.
- It is implemented by a set of properties called attributes, along with its values and the links it may have on other objects.



Object's Behavior

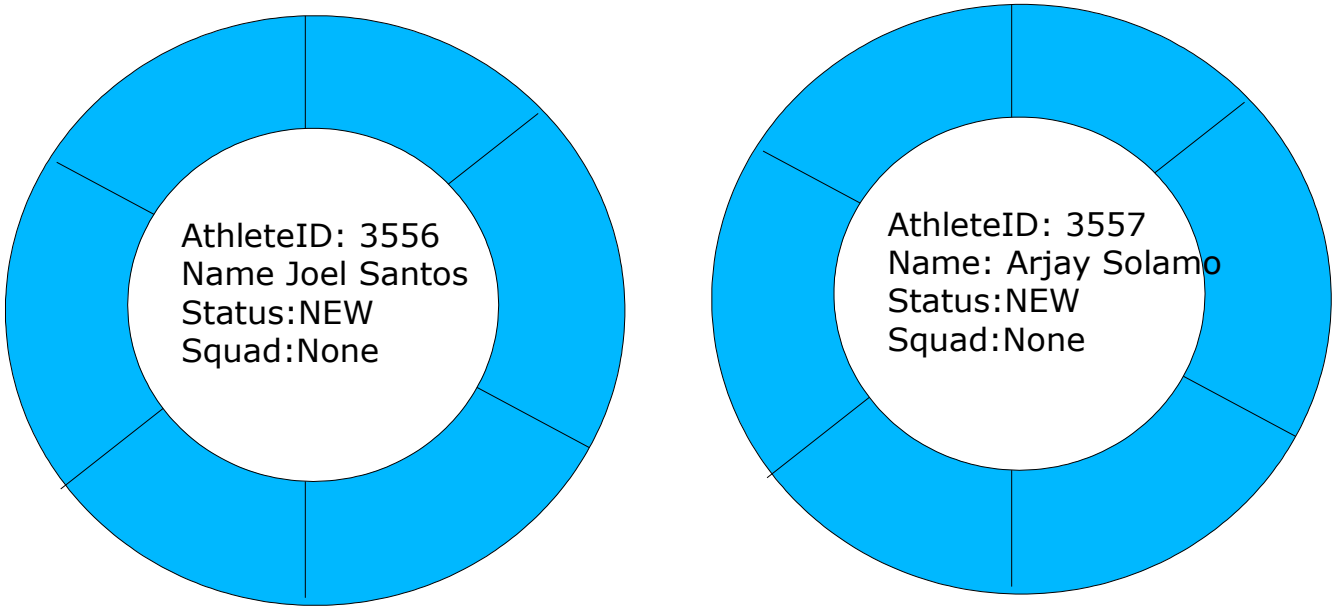
- It determines how an object acts and reacts.
- It is represented by the operations that the object can perform.



Joel Santos

Object's Identity

- Although two objects may share the same state (attributes and relationships), they are separate, independent objects with their own unique identity.



Four Basic Principles of Object-orientation

- Abstraction
- Encapsulation
- Modularity
- Hierarchy

Abstraction

- Abstraction is a kind of representation that includes only the things that are important or interesting from a particular point of view.
- It is the process of emphasizing the commonalities while removing distinctions.
- It allows us to manage complexity systems by concentrating on the essential characteristics that distinguish it from all other kinds of systems.
- It is domain and perspective dependent.

Sample Abstraction

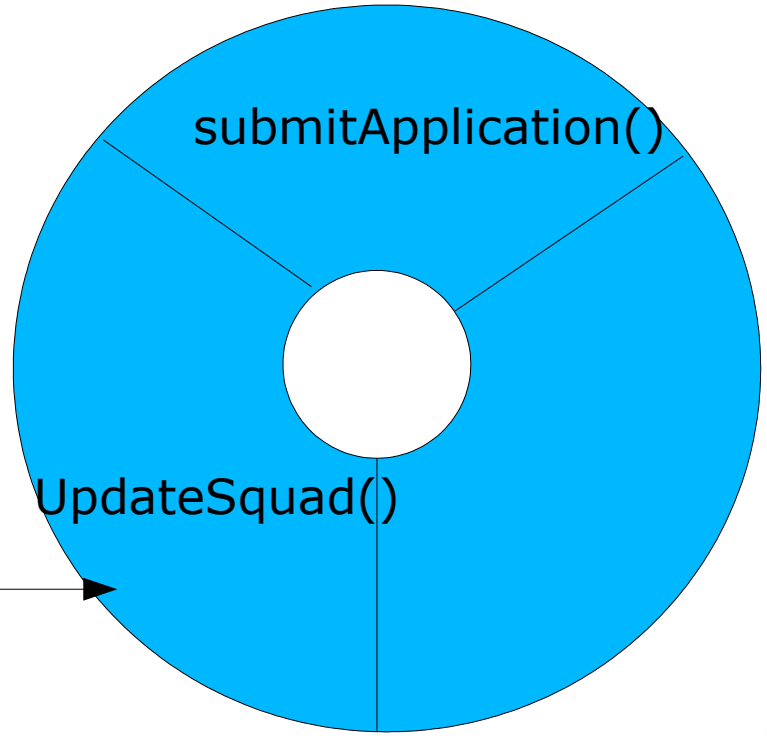
- An applicant submits a club membership application to the club staff.
- A club staff schedules an applicant for the mock try-outs.
- A coach assigns an athlete to a squad.
- A squad can be a training or competing squad.
- Teams are formed from a squad.

Encapsulation

- Encapsulation localizes features of an entity into a single blackbox abstraction, and hides the implementation of these features behind a single interface.
- It is also known as ***information-hiding***; it allows users to use the object without knowing how the implementation fulfils the interface.
- It offers two kinds of protection: it protects the object's state from being corrupted and client code from changes in the object's implementation.

Encapsulation Illustrated

- Juan de la Cruz needs to change his year level.
- The key is in the ***message interface***.

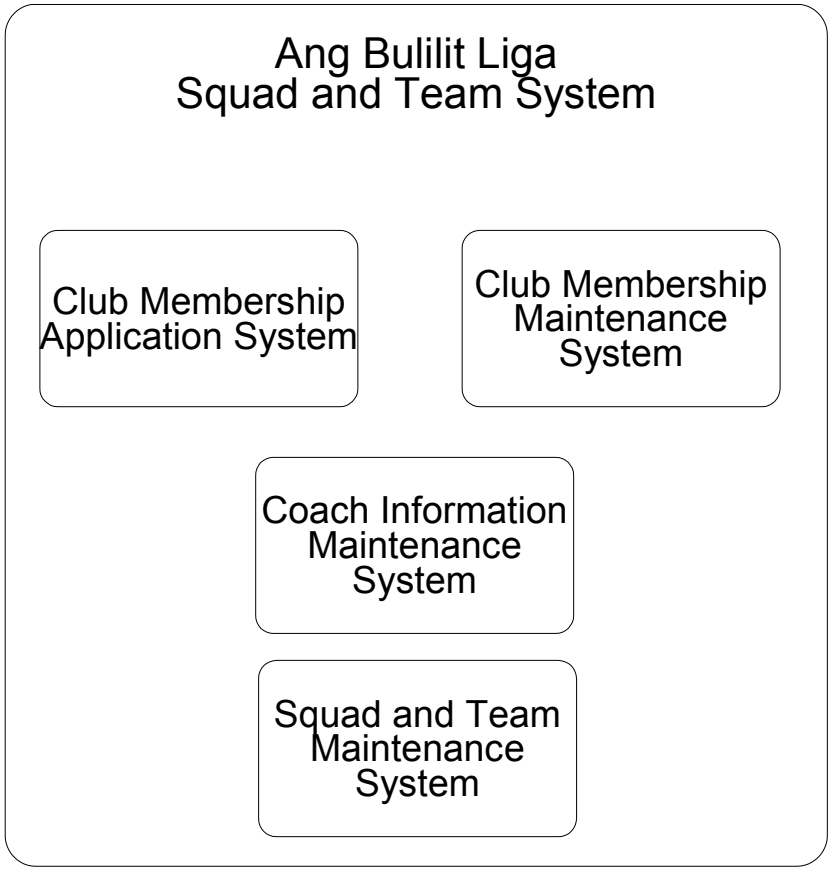


updateSquad("Training")

Modularity

- Modularity is the physical and logical decomposition of large and complex things into smaller and manageable components that achieve the software engineering goals.
- It is about breaking up a large chunk of a system into small and manageable subsystems. The subsystems can be independently developed as long as their interactions are well understood.

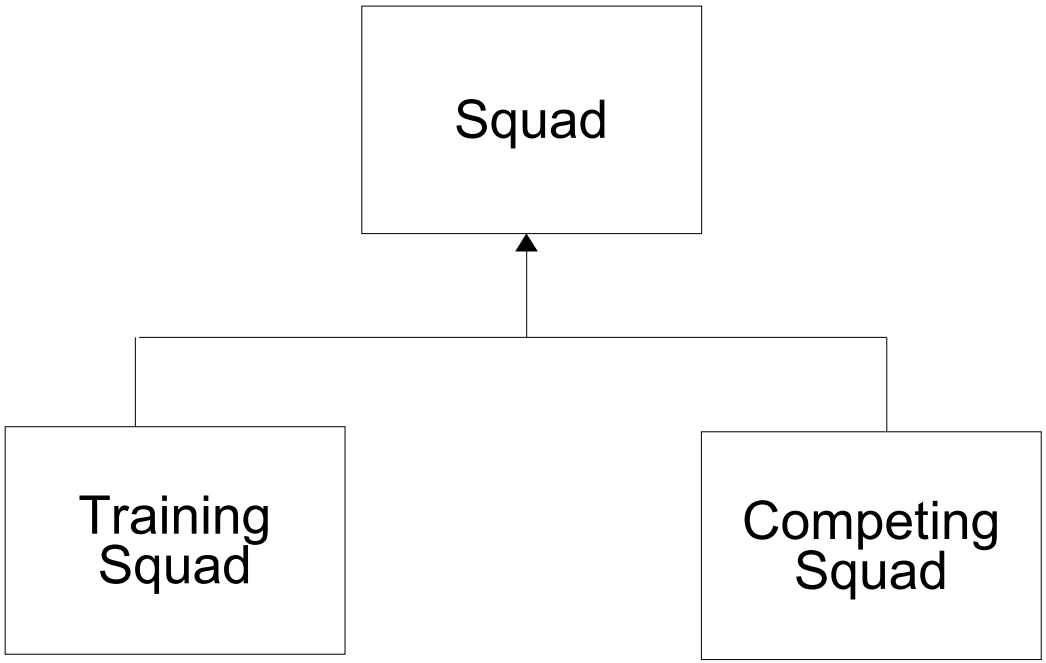
Modularity Illustrated



Hierarchy

- Any ranking or ordering of abstractions into a tree-like structure.
- Kinds of Hierarchy
  - Aggregation
  - Class
  - Containment
  - Inheritance
  - Partition
  - Specialization
  - Type

Hierarchy Illustrated

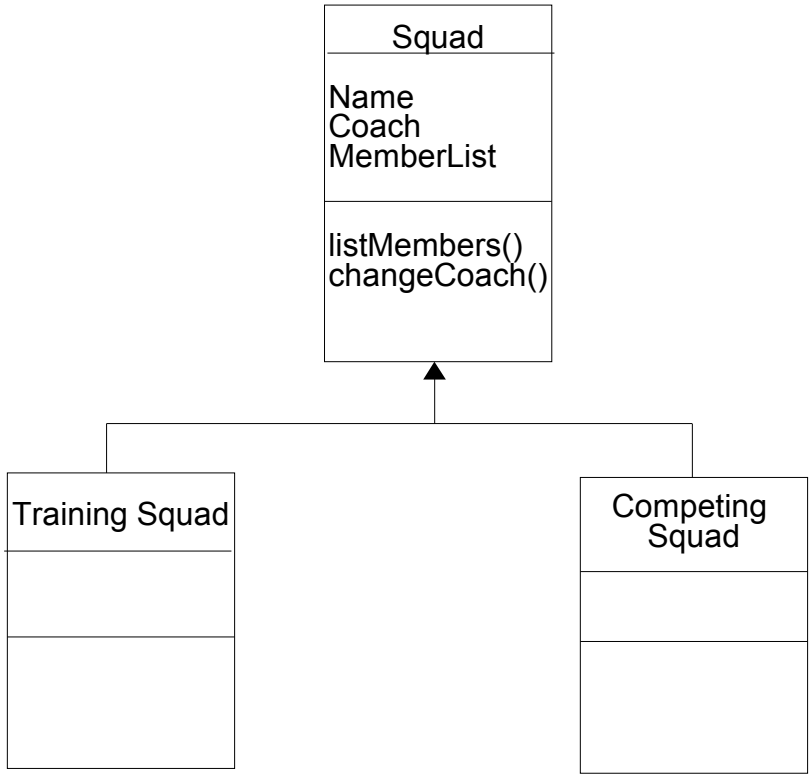


Generalization

- It is a form of association wherein one class shares the structure and/or behavior of one or more classes.
- It defines a hierarchy of abstractions in which a subclass inherits from one or more superclasses.
  - Single Inheritance
  - Multiple Inheritance
- It is an ***is a kind of*** relationship.

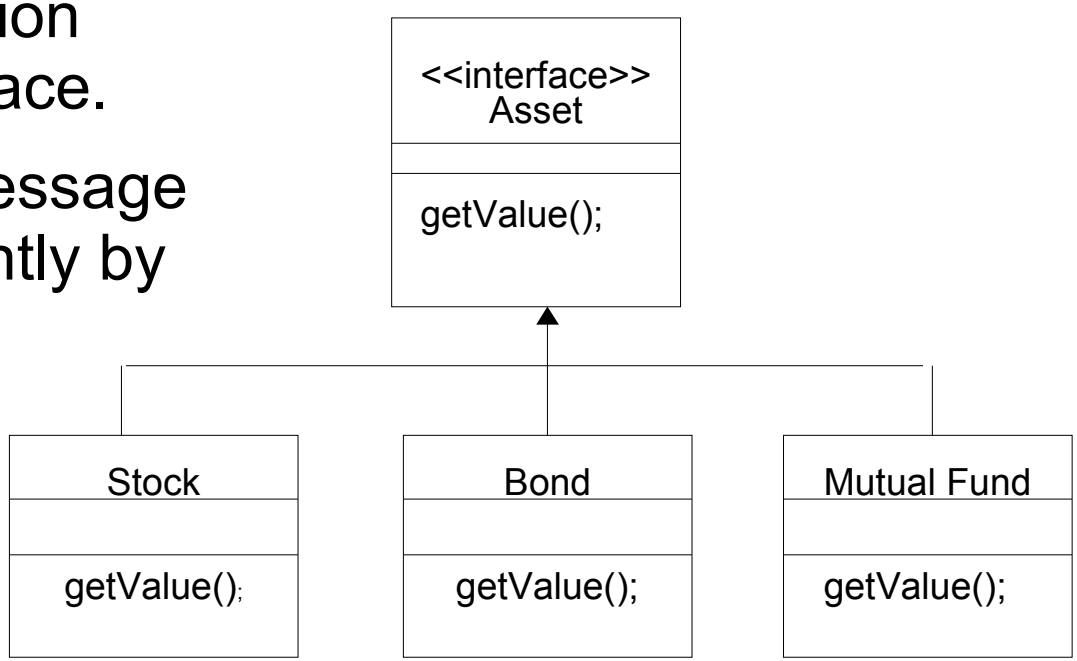
Inheritance

- It is a mechanism by which more-specific elements incorporate the structure and behavior of more-general elements.
- A class inherits attributes, operations and relationship.



Polymorphism

- It is the ability to hide many different implementation behind a single interface.
- It allows the same message to be handled differently by different objects.

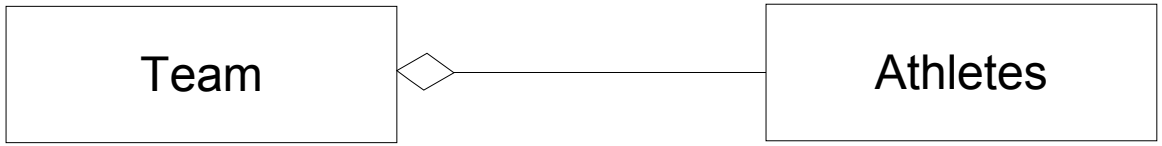


Interface

- It formalizes polymorphism. It defines polymorphism in a declarative way, unrelated to implementation.
- It is the key to the ***plug-n-play*** ability of an architecture.

Aggregation

- It is a special form of association that models a whole-part relationship between an aggregate (whole) and its parts.



Summary

- Object Technologies
- Objects
  - Object's State
  - Object's Behavior
  - Object's Identity
- Four Basic Principles of Object-orientation
  - Abstraction
  - Encapsulation
  - Modularity
  - Hierarchy