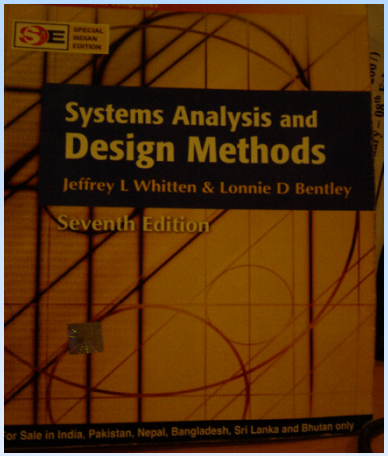
Object Oriented Systems Analysis and Design

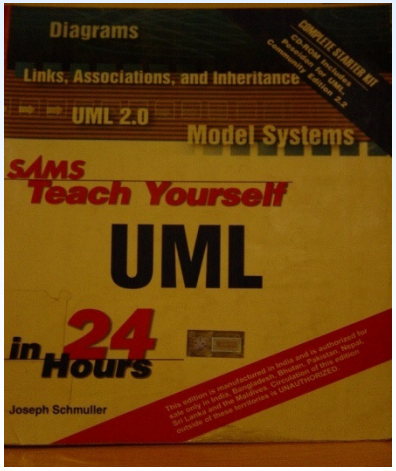
IT 3105

References

Ref : 1. System Analysis and DESIGN METHODS **By** Jeffrey L Whitten & Lonnie D Bentley
ISBN 0-07-063417-3
(7th Edition)

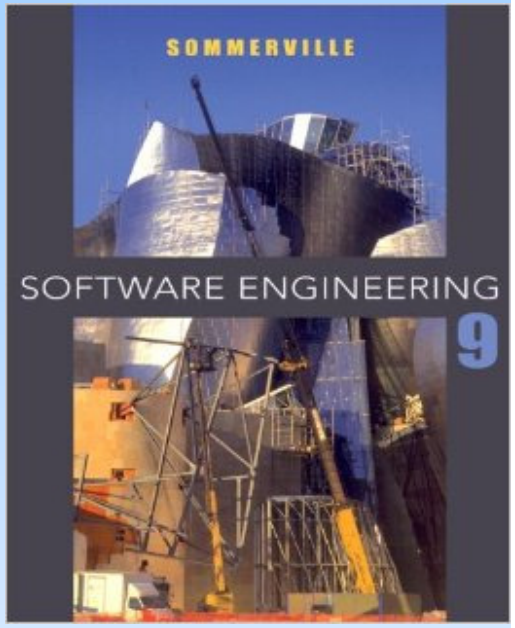
2. SAMs Teach Yourself UML in 24 Hours,
Joseph Schmuller, 3rd Edition,
ISBN 81-297-0609-1,
Pearson Edu.,2004





UCSC

References



- Ref 3 : Software Engineering, Ian Somerville, 9th edition ,
  ISBN 978-81-317-6216-5 ,Pearson ,
  2011,
- Ref4: http://en.wikipedia.org/wiki/Rational_Unified_Process
- Ref5: http://www.uml-diagrams.org

Exploring OOAD and Development

- To write any piece of software you need to do three things.
  - OO Analysis - **Understand – (What)**
  - OO Design - **Plan – (How)**
  - OO Programming - **Build**

Traditional Methodologies vs Object Oriented Methodologies

- During Analysis most likely to change ,
  - *Processes , External Interfaces, attributes e*tc.
- Most stable aspects,
  - *Classes and Objects*

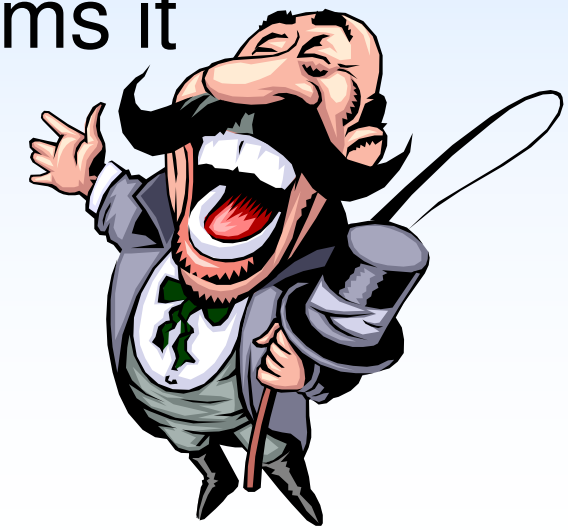
Real world concepts that are to be modeled

- Object oriented approach focuses on the more stable aspect of the system.

Introduction…

Object Oriented Methodologies

- Object Orientation has taken the software world by storm.
- As a way of creating programs it has number of advantages.



Introduction…

Object Oriented Methodologies

- It promotes the component- based approach to software development.
  - First create a system by creating a set of objects.
  - Then, you can expand the system by adding capabilities to components you have already built *or* by adding new components.
  - Finally, you can reuse the objects you created for the system when you built a new system. This will substantially reduce the system development time.

Introduction…

Object Oriented Methodologies

- UML (Unified Modeling Language) plays an important role in Object Orientation.
- UML allows you to built easy to use and easy to understand models of objects so that programmers can easily write software.
- Object Orientation depends on a few fundamental principles.

System concepts for object modeling

- A problem can be divided into subjects.
- Subjects are equivalent to sub systems in traditional methodologies.
- Subjects are used as logical divisions for development of large systems
- Each subject is generally considered as a problem domain

Subjects ■eg. ***Airport Management System***

Radar control *Baggage Handling* *Passenger Processing*

UCSC

System concepts for object modeling

- We understand the real world through ideas that are organized in to recognizable *patterns* and *concepts.*
- *Concepts* are divided into following types :
  - Tangible - *book*
  - Roles - *doctor*
  - Relational - *marriage*
  - Intangible - *time, quality*
  - Judgmental - *good pay*
  - Event - *Purchase, loan*

System concepts for object modeling

- In the real world interaction of these concepts allow complex operations to be performed.

Library System

Core Concepts - *Lending, Returning of books, Book registration (Scenarios)*

Concepts - *Borrower, Copy, Book (Classes)* ← **Category**

*** He knows certain facts -** eg. Name address etc.

*** Also perform certain activities**

Instance of a Human Being Concept

Activities normally involves interaction with other individuals or institutions.

ucsc

System concepts for object modeling

- Classes
  - Represent the concepts that are to be modeled.
  - Initial Step in OOA is the identification of *classes (Problem Domain)*.
  - To ease this process identify *Scenarios –Lending books, returning of books , registration of Books* in a library system
  - Classify type of expected users of the system (Librarian)
  - *Classes* in a library system - *Book, Copy, Borrower etc.*

System concepts for object modeling

- Objects
  - **Something that is or is capable of being seen, touched, or otherwise sensed and about which users store data and associate behavior**
  - **Types of objects**
    - **Person – e.g. employee, customer, instructor, student**
    - **Place – e.g. warehouse, building, room, office**
    - **Thing – e.g. product, vehicle, computer, videotape**
    - **Event – e.g. an order, payment, invoice, application**
    - **Sensual – e.g. phone call, meeting**

System concepts for object modeling

- Attributes and Services
  - By examine the descriptions/documents given by the users, event type concepts (Scenarios) can be identified
  - Attributes : The data that represents characteristics of interest about an object.
  - Services : The set of things that an object can do and that correspond to functions that act on the object's data or attributes.

System concepts for object modeling

- **Attributes and Services**

eg. Library System

Basic Scenarios → *Loan of Books*

Basic Scenarios → *Return of Books*

Basic Scenarios → *Reservation of Books*

Each Scenario → Pool of Classes

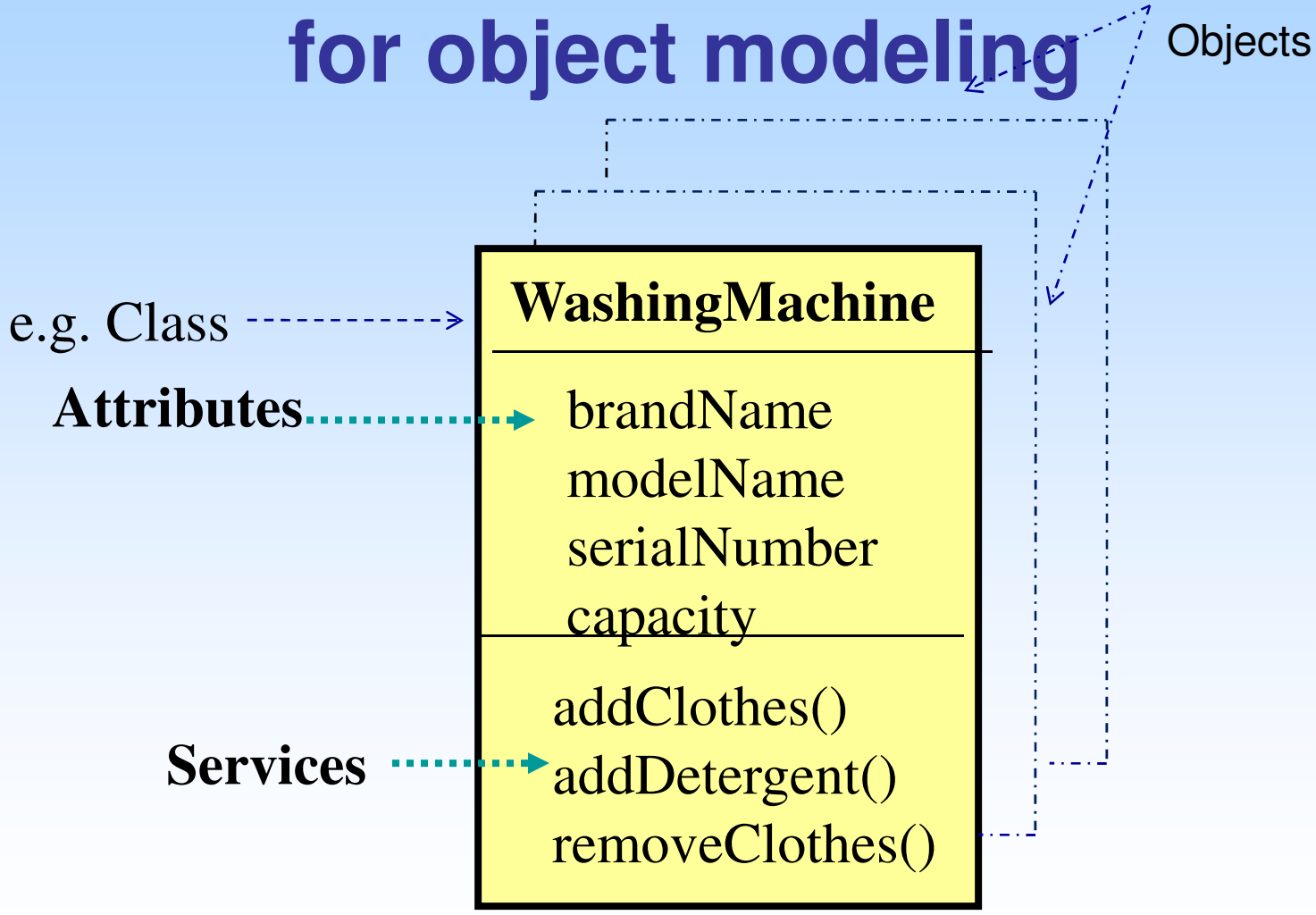
Pool of Classes → Each class offer

* *Various* ***Services***

* *Contains information reflected as* ***Attributes***

ucsc

System concepts for object modeling



System concepts for object modeling

- A principle used to derive *attributes* and *services*

  * What the specific concept knows ?
    - *attributes*
  * What are the responsibilities of a class ?
    - *services*

Services operate when they are involved through message connections.

Knows how to draw a rectangle

but he will not draw unless somebody ask him to draw

System concepts for object modeling

- Abstraction
  - A mechanism to reduce and filter out details so that one can focus on a few concepts at a time. Focus on Essentials
  - Filter out an object's properties and operations until just the ones you need are left.

    Ignore the irrelevant. Ignore the unimportant.

System concepts for object modeling

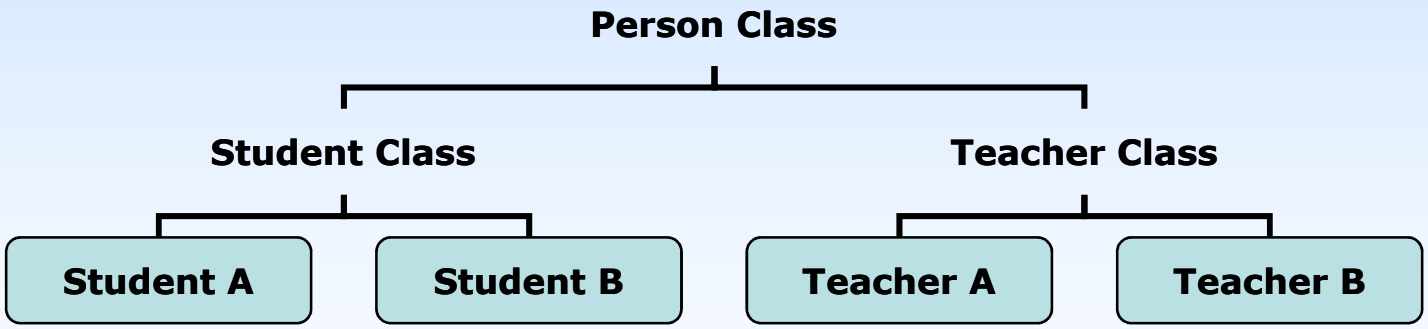
- Inheritance
  - The concept wherein methods and/or attributes defined in an object class can be inherited or reused by another object class.

    e.g. some individuals in the room might be classified as STUDENTS and TEACHERS. Thus, STUDENT and TEACHER object classes are members of the object class PERSON

System concepts for object modeling

- Inheritance

  e.g. Cont…

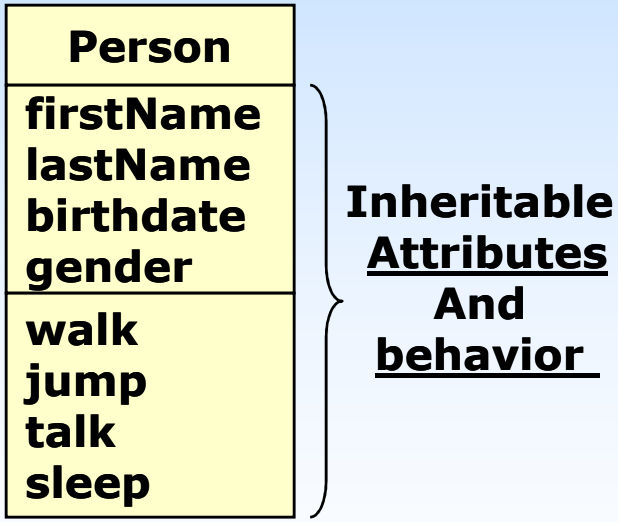


System concepts for object modeling

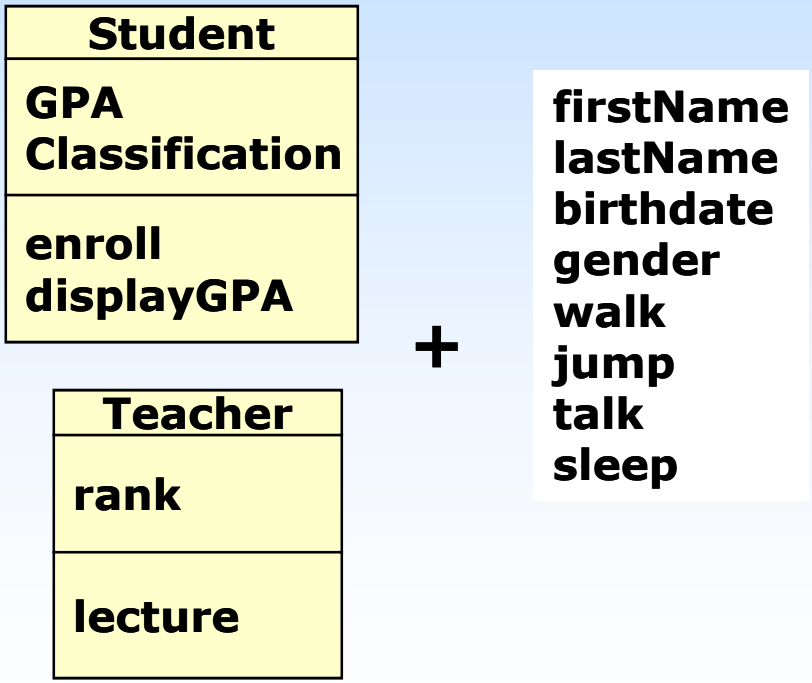
- Generalization / Specialization
  - A technique wherein the attributes and behaviors that are common to several types of object classes are grouped / abstracted into their own class called a supertype.
  - The attributes and methods of the supertype object class are then inherited by those object classes (subtype)
  - Sometimes abbreviated as gen/spec.

System concepts for object modeling

Generalization



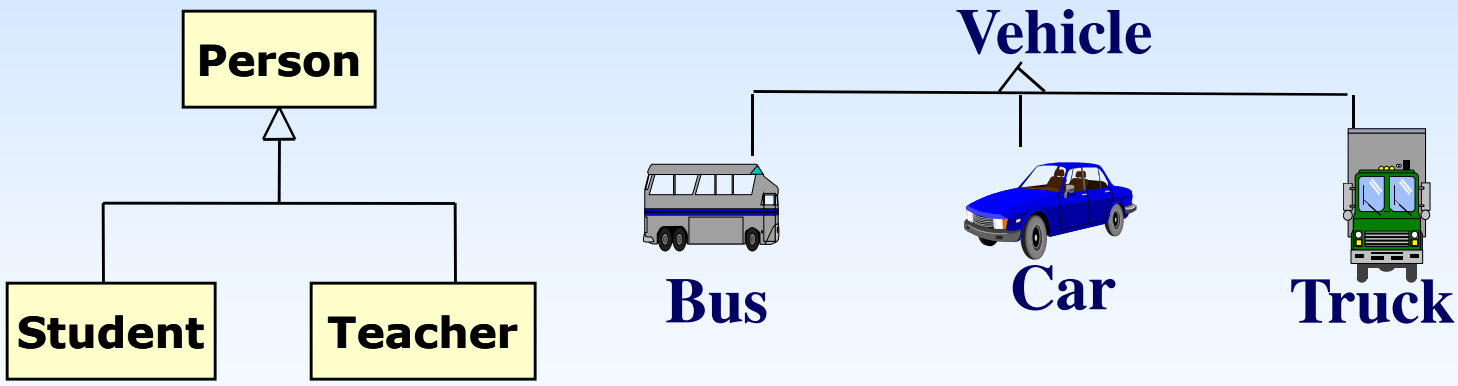
Specialization



UCSC

System concepts for object modeling

- Generalization / Specialization



*** Specialized classes inherits from the parent class**

System concepts for object modeling

- Polymorphism
  - It allows different forms of the same service to be defined.
  - Sometimes an operation has the same name in different classes.

    eg. You can open a door, you can open a window, and you can open a newspaper, a bank account or a conversation. In each case, you are performing a different operation.

System concepts for object modeling

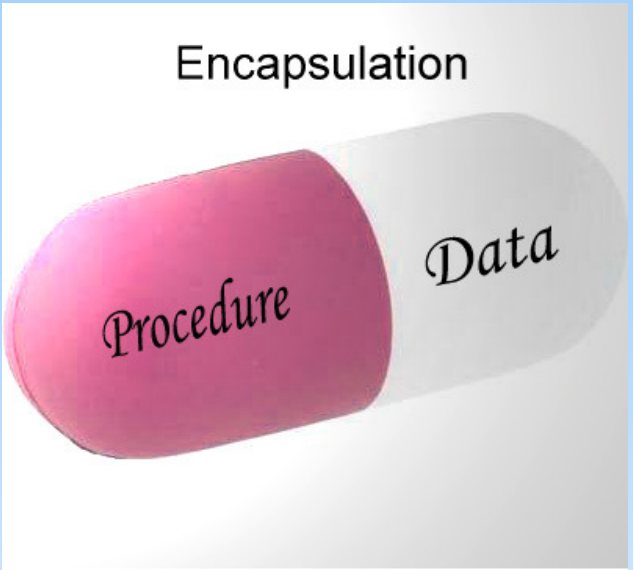
- Polymorphism
  - In object-orientation, each class “knows” how that operation is supposed to take place.
  - This is polymorphism.
  - *Related terms used in OO programming language*

    Function Overloading,

    Operator Overloading,

    Method Overriding

System concepts for object modeling



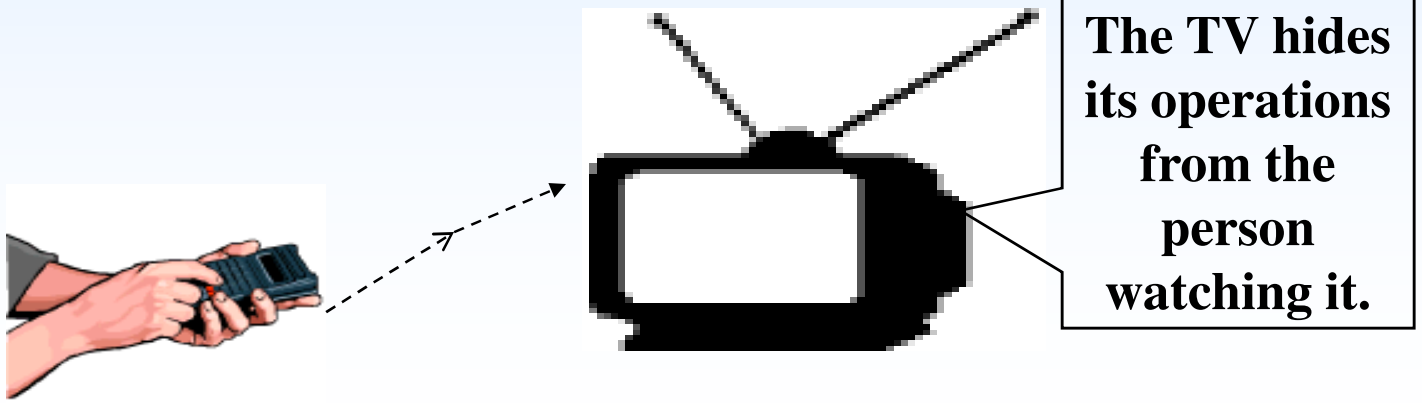
- Encapsulation
  - **Packaging of several items together into one unit (both attributes and behavior of the object), Also protects the contents.**
  - **The only way to access or change an object's attribute is through that object's specific behavior.**
  - **Objects *encapsulates* what they do.**
    - That is, they hide the inner workings of their operations
      - **from the outside world**
      - **and from other objects**

System concepts for object modeling

- Encapsulation

  When an object carries out its operations, those operations are hidden.

E.g. When most people watch a television show,
- they usually don't know or care about the complex electronics that sit in back of the TV screen
- or the operations that are happening.



System concepts for object modeling

- Encapsulation

  Why is this important?

  – In the software world, encapsulation helps cut down on the potential for bad things to happen.

  – In a system that consists of objects, the object depends on each other in various ways.

  – If one of them happen to malfunction, software engineers have to change it in some way.

  – Hiding its operations from other objects means it probably won't be necessary to change those other objects.

System concepts for object modeling

- Encapsulation
  - Turning from software to reality, you see the importance of *encapsulation*.

eg. Your computer monitor, hides its operations from your computers CPU.

When something goes wrong with your monitor, you either fix or replace it.

You probably won't have to fix or replace the CPU along with it.

System concepts for object modeling

- Encapsulation
  - Encapsulation is also called *information hiding*. (An object hides what it does from other objects and from outside world)
  - But an object does have to present a “face“ to the outside world, so you can initiate those operations.

    eg. A TV, has a set of buttons either on the TV or on a remote.

    The TV’s buttons are called *interfaces*.

System concepts for object modeling

- Encapsulation
  - *Traditional methodologies*
    - Main program is the focus.
    - It brings in data and perform processing.
  - *OO Methodology*
    - Each object contains
      - Information it is responsible for
      - Services it supports

System concepts for object modeling

- Encapsulation
  - –Encapsulation is done through the definition of *region of access*.
  - –*Region of access* defines the accessibility of the *services* or *attributes* in a class.
  - –In C++ (OO programming Language)
    - 3 types of access regions: *private, public, protected*
  - –*In Java*
    - *Access modifiers: default, public, private, protected,*
    - *Static modifier, final modifier, synchronized modifier, native modifier*

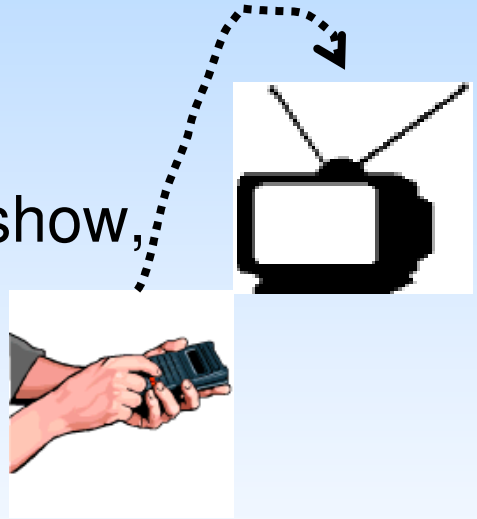
System concepts for object modeling

- Message Passing
  - In a system, objects work together.
  - They do this by sending messages to one another.
  - One object sends another a message to perform an operation.
  - The receiving object performs that operation.

    eg. A TV and a Remote

System concepts for object modeling

- Message Passing
  - When you want to watch a TV show,
    - You hunt round for a remote,
    - Settle into your favorite chair and
    - Push the *On* Button.
  - What happens?
    - The remote object sends a message (literally!) to the TV object to turn itself on.
    - The TV object receives the message.
    - It knows how to perform the turn-on operation, and turns itself on.



System concepts for object modeling

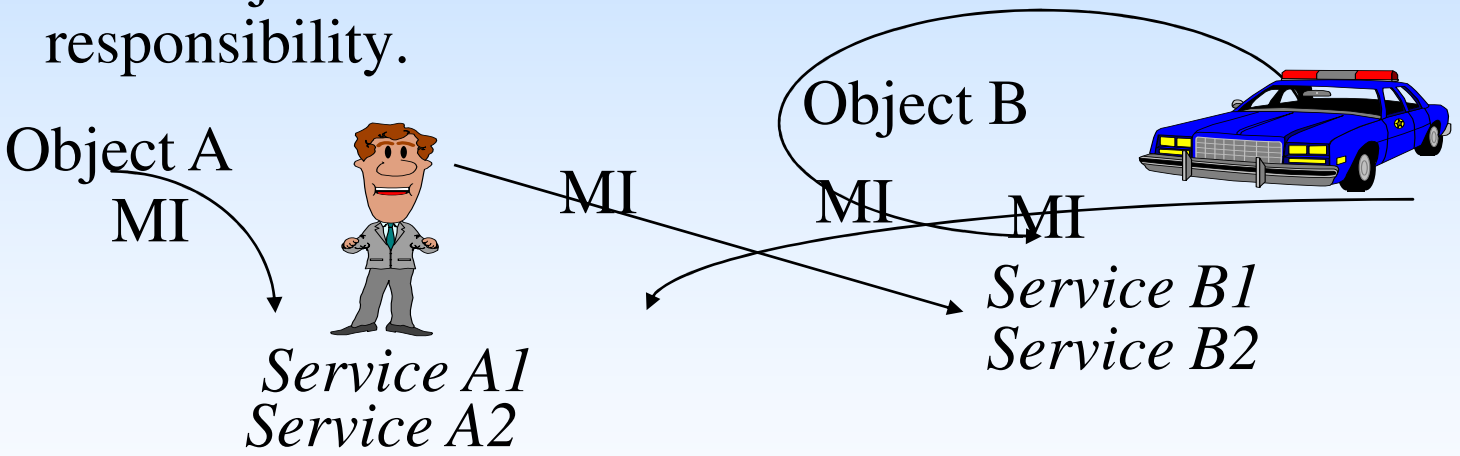
- Message Passing
  - When you want to watch a different channel,
    - You click the appropriate button on the remote,
    - Remote object sends a different message *change channel* to the TV object.
  - The remote can also communicate with the TV via other messages for *changing the volume, muting the volume etc.*.

System concepts for object modeling

- Message Passing
  - Let's go back to interfaces for a moment.
  - Most of the things you do from the remote, you can also do by *getting out of the chair*, *going to the TV*, and *clicking buttons on the TV.*
  - The interface the TV presents to you is obviously not the same interface it presents to the remote.

System concepts for object modeling

- Each object offers services within the realm of its responsibility.



* Objects use other objects services through *message passing*

* An object may even call its own services

System concepts for object modeling

- Relationships
  - A natural business association that exists between one or more objects and classes

    e.g. You interact with a text book by reading it, with a telephone by using it, People interact with each other by communicating with them.

System concepts for object modeling

- Association
  - **When you turn on your TV, in object oriented terms, you are in an *association* with your TV.**
  - **An association is unidirectional (one way) or bi-directional (two way).**

    eg. *is married to*
  - **Some times an object might be associated with another in more than one way.**

    Gihan *is a co-worker of* Damith

    Gihan *is a friend of* Damith

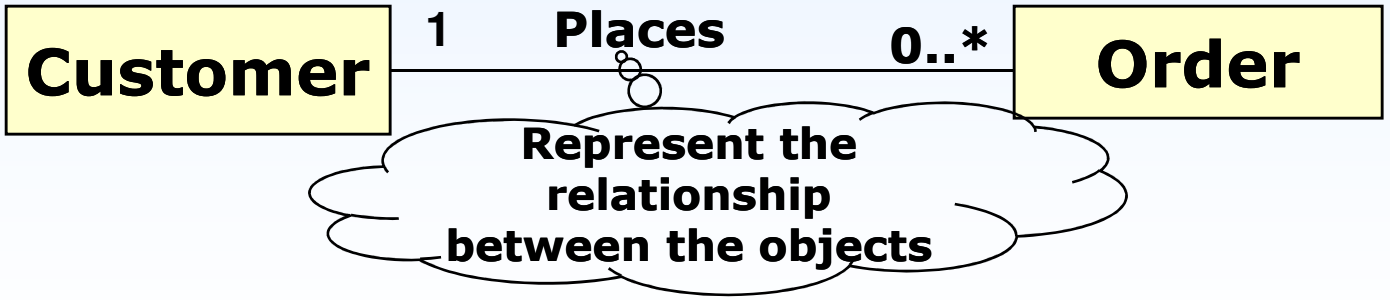
System concepts for object modeling

- Association

  e.g.

  A CUSTOMER PLACES zero or more ORDERS

  An ORDER IS PLACED BY one and only one CUSTOMER



System concepts for object modeling

- Multiplicity
  - The minimum and maximum number of occurrences of one object class for a single occurrence of the related object class.

    e.g. Exactly one -> **1**

    Zero or 1 -> **0..1**

    Zero or more -> **0..*** or ***

    1 or more -> **1..***

    Specific range -> **7..9**

System concepts for object modeling

- *Aggregation*
  - A relationship in which one larger "whole" class contains one or more smaller "parts" classes. Conversely, a smaller "part" class is part of a "whole" larger class.

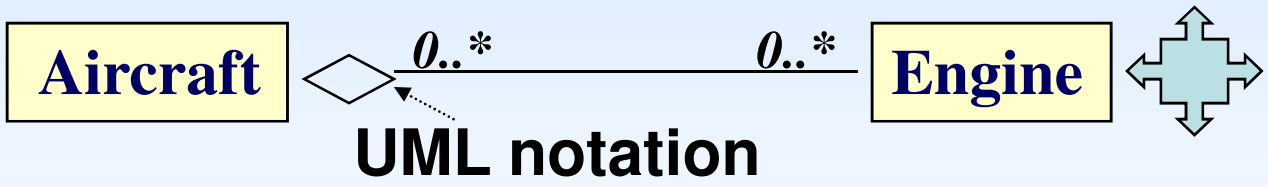
    e.g. *A club* – a club is made up of several club members

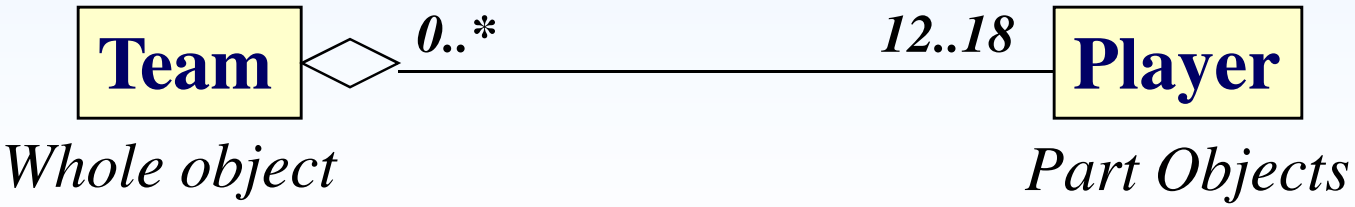
    A computer – a computer contains a case, CPU, motherboard, power supply …etc.

System concepts for object modeling

- *Aggregation* ***(Removed in UML 2.x)***

  some more examples…





System concepts for object modeling

- *Composition*
  - An aggregation relationship in which the "whole" is responsible for the creation and destruction of its "parts".
  - If the "whole" were to die, the "part" would die with it.
  - A stronger form of aggregation.
    - The relationship between club and club member would not be composition, because members have a life out-side the club and can, belong to multiple clubs.

System concepts for object modeling

- *Composition*
  - Drawn with a filled diamond.

Each "part" can belong to only one "whole", therefore, multiplicity needs to be specified only one for the "whole"

Components will live and die with the whole object