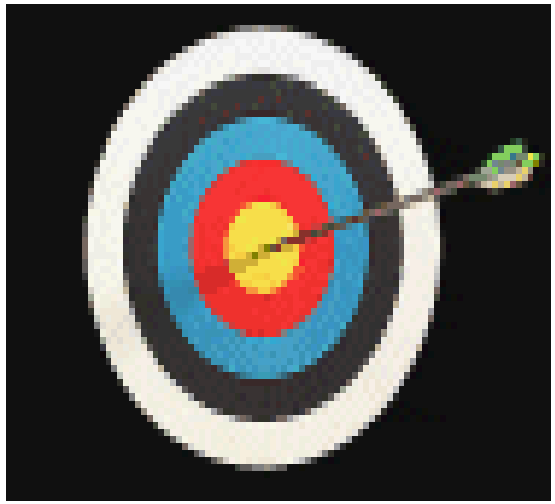
UML

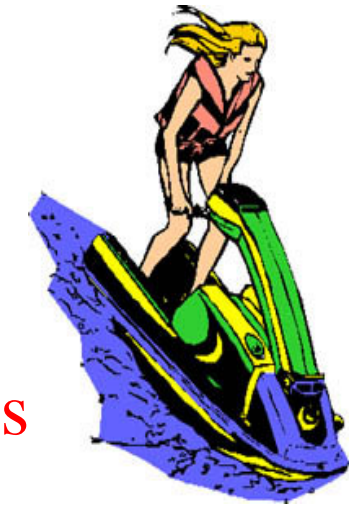
Understanding it.
Exploring it.
Exploiting it !!!



Raghavendra(raghu)
Hughes s/w Systems.

Objects

- Encapsulation
  - Properties
  - Methods
- Inheritance
- Aggregation
- Relationships
- Interactions
- **Interfaces**



- Creation
- Life Time management
- Binary Representation
- ‘Talk’

Dann klappt's auch mit dem Reifenwechsel.

Modeling

Booch Method

Jacobson

OMT-2 (Rumbaugh)

UML

Goals of UML

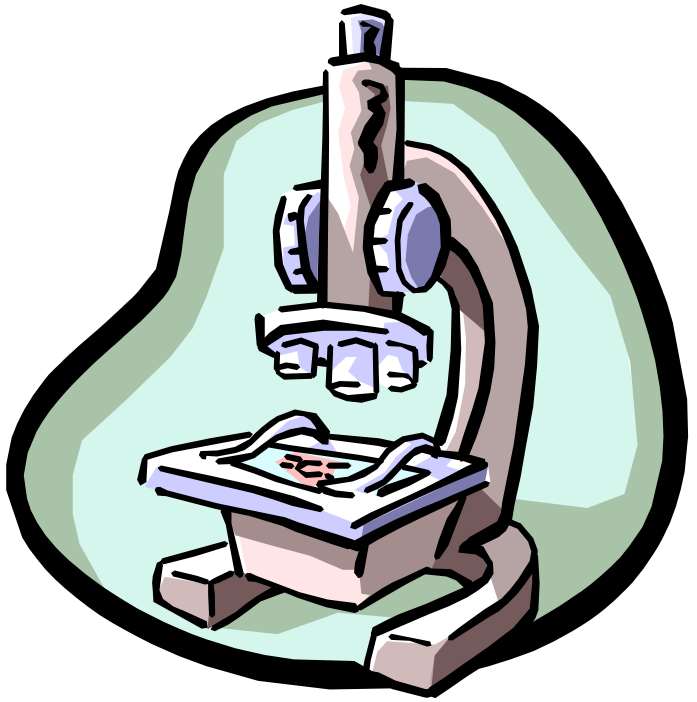
- To model systems using OO concepts
- Explicit coupling between conceptual and executable artifacts
- To address issues of Scale
- Usable by both Humans and machines

UML

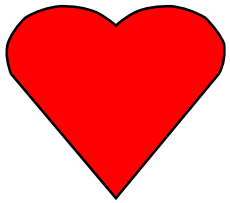
- Views
- Diagrams
- Model elements
- General Mechanisms

Views

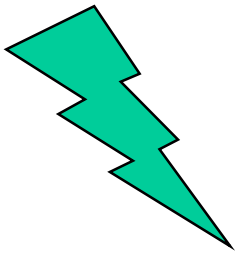
- Use Case view
- Logical view
  - static
  - dynamic
- Component view
- Concurrency view
- Deployment view

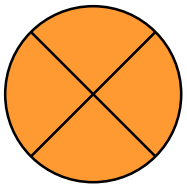


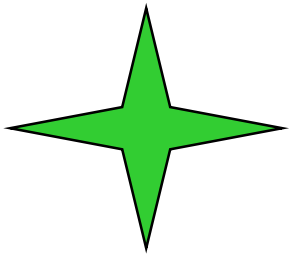
Diagrams

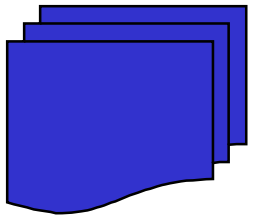


- Actual graphs of model elements
- Generally allocated to a view
- Can be a part of more than one view

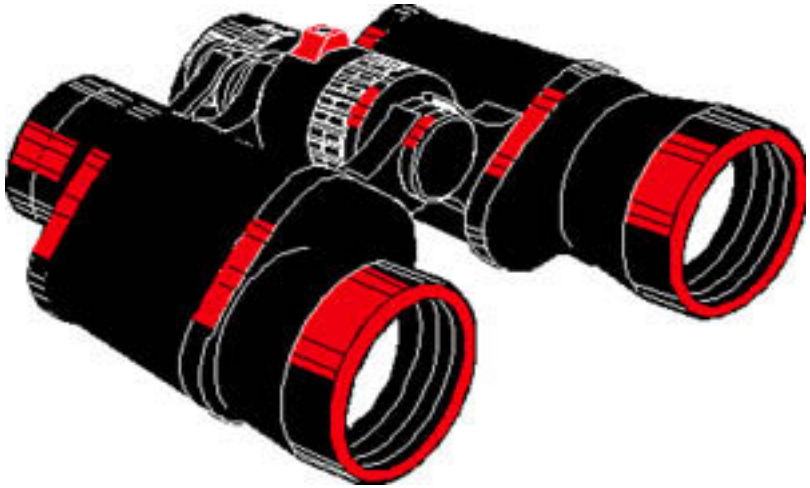








Use Cases

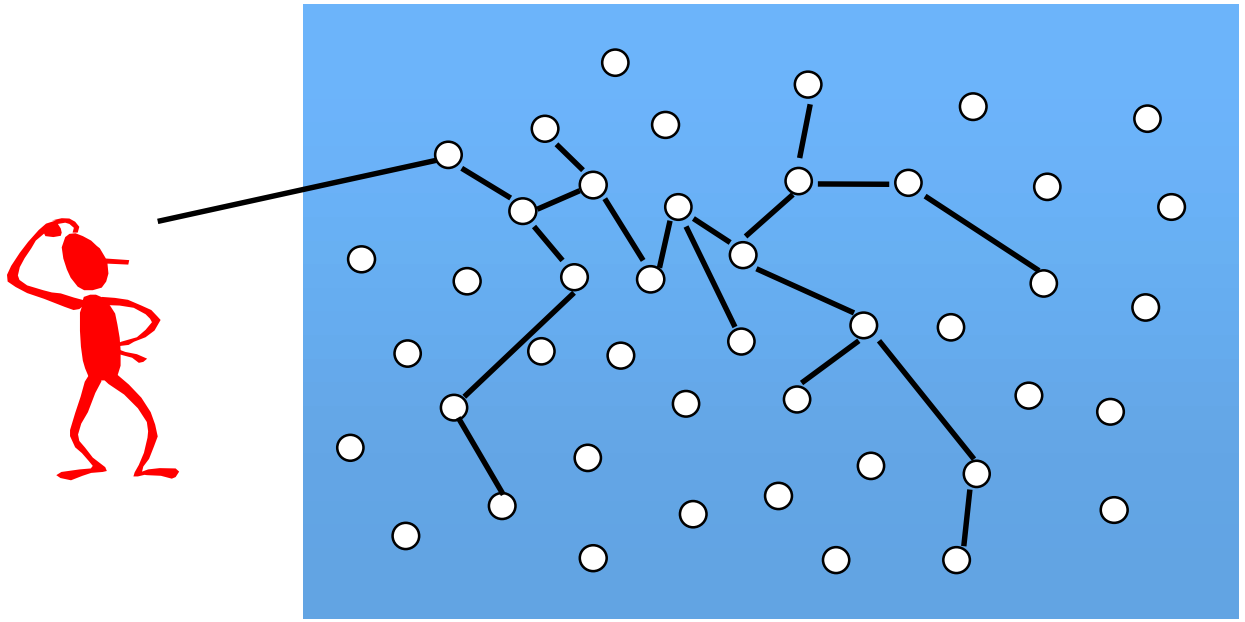


Marriage Bureau

Who likes Chocolates ???

Use Cases

Use Case Analysis is a technique to capture business process from user's perspective



Describing a usecase

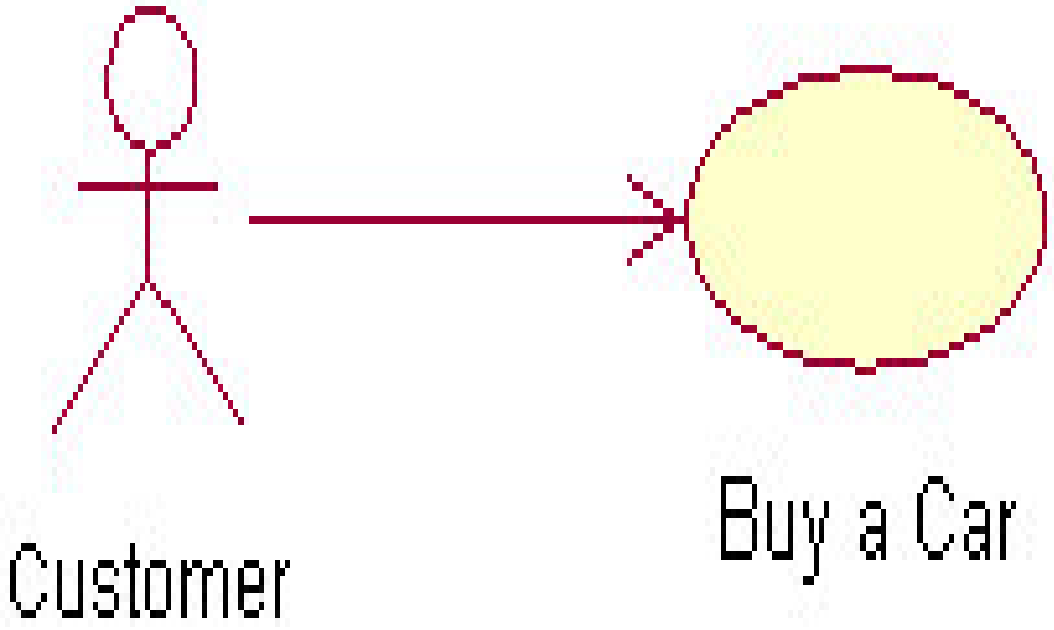
- Generally done as plain text
- Activity diagrams can also be used

Use Cases Jargon !

- Actors
  - Primary
  - Secondary
  - Tertiary
- Interactions
  - simple
  - compound
- Scenario
- Sequence
- Goals and Responsibilities

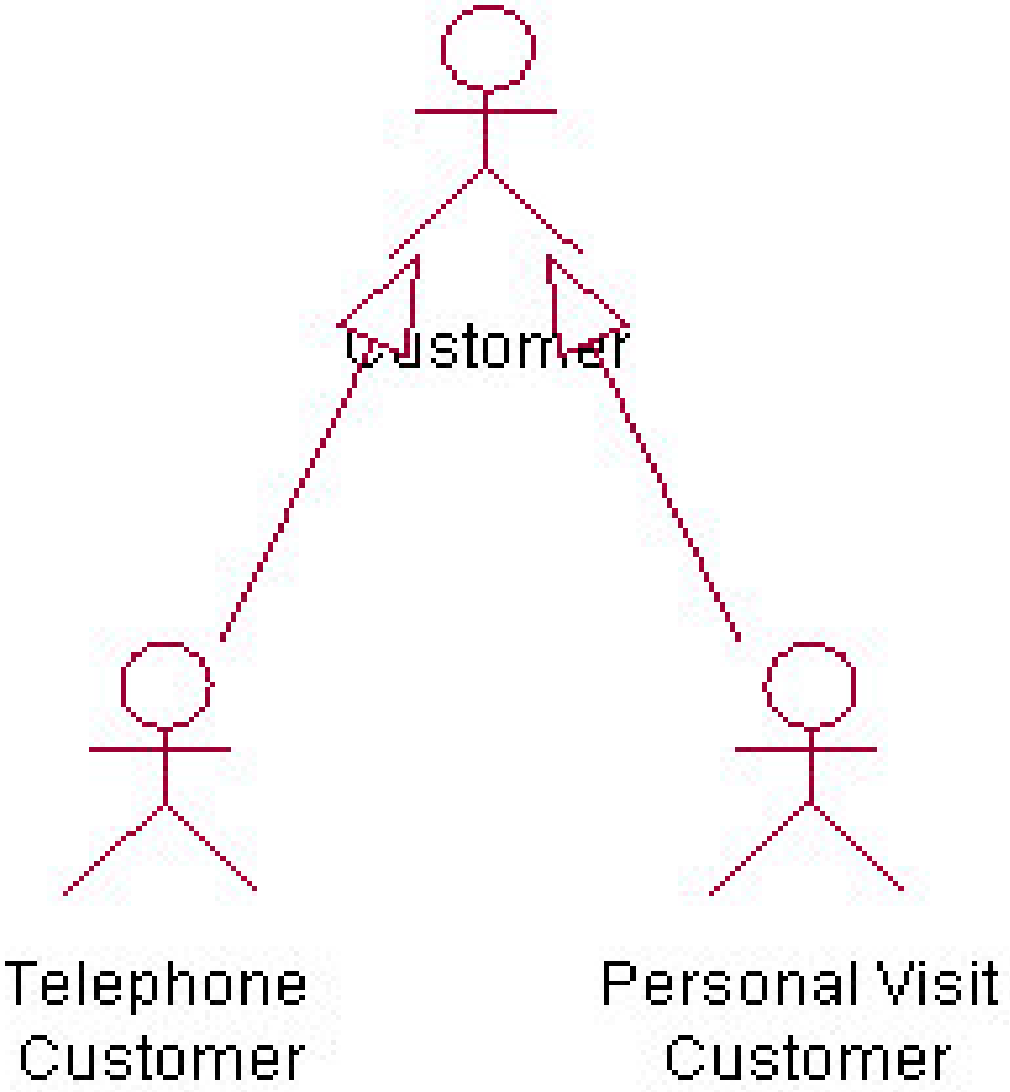
Relationships in UseCases

- UseCase-Actor
  - Association



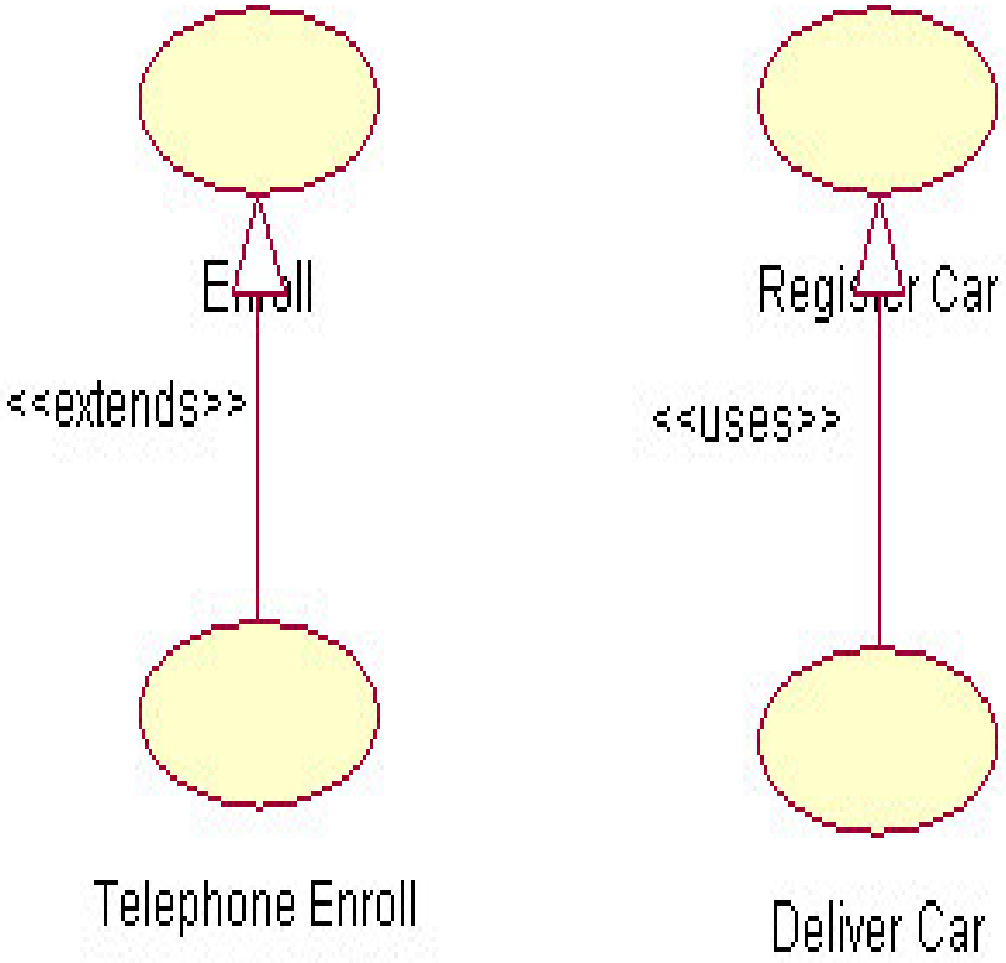
Relationships in UseCases

- Actor-Actor
  - Generalizes



Relationships in UseCases

- UseCase-UseCase
  - extends
  - uses



Tips !

How much do we need to dress them up ?

- Business Use cases
- System Use Cases

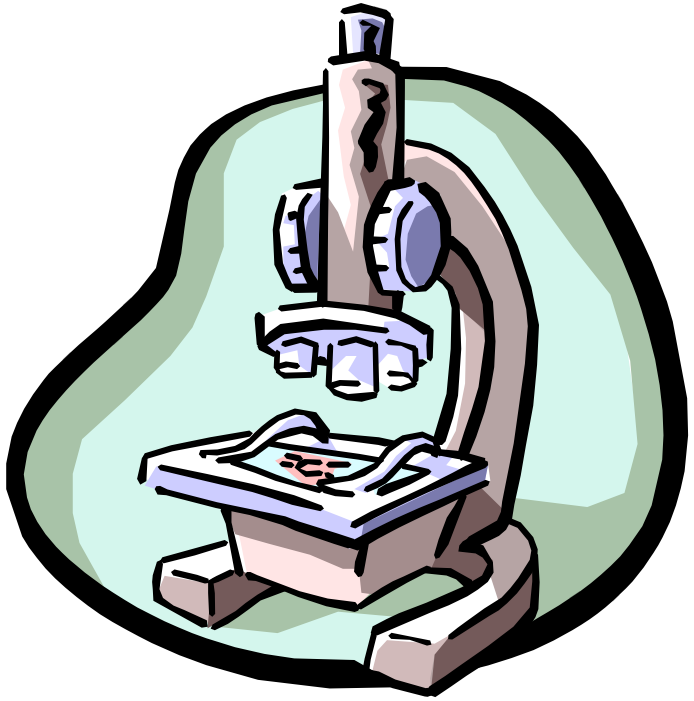
What is the Scope of your Use Case ?

- Organization
- System Scope

UseCases help in

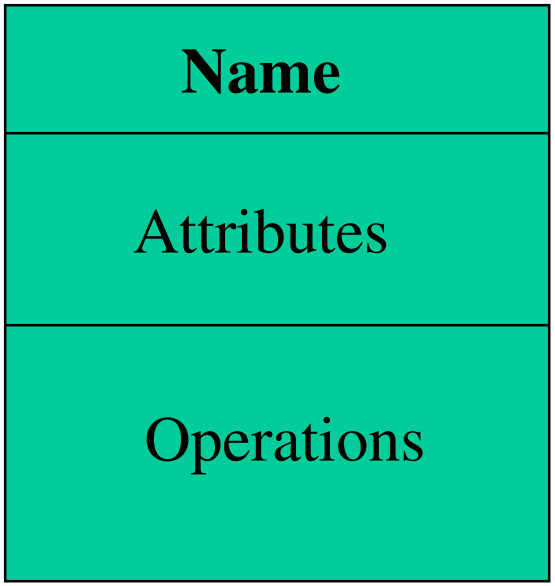
- Requirements
- Design
- Testing
- Planning

Static Modeling
(Class Diagrams)



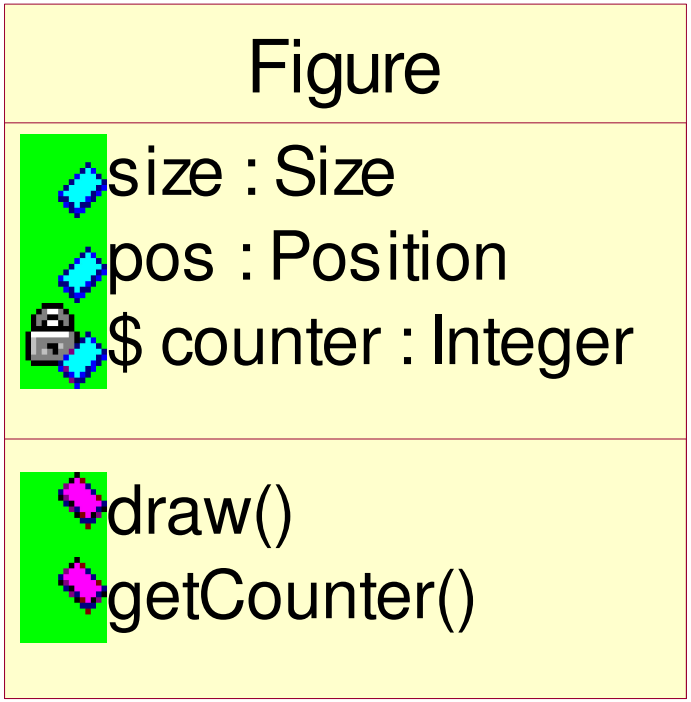
Class Diagrams

- Static view of the System
- Classes and their relationships
- Object diagrams slightly vary



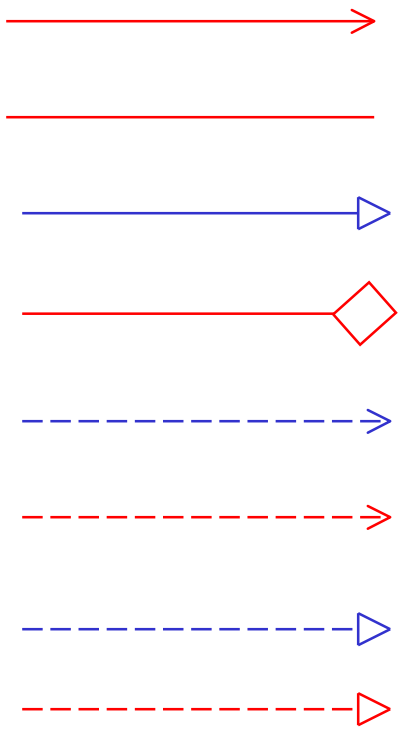
Example of a class

Figure
+ size:Size + pos:Position - counter:Integer
+draw() +getCounter() : Integer

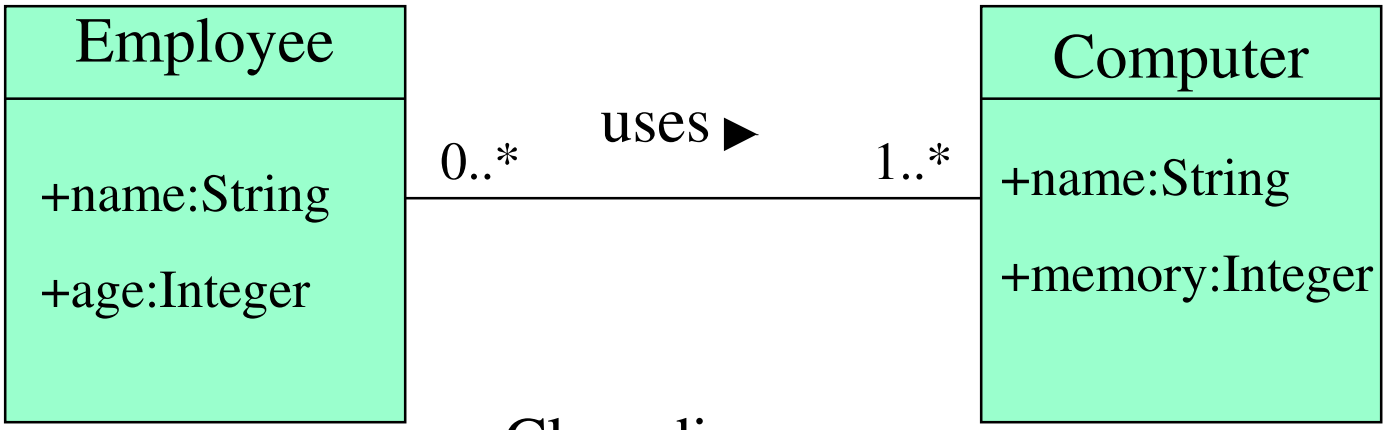


Relationships between Classes

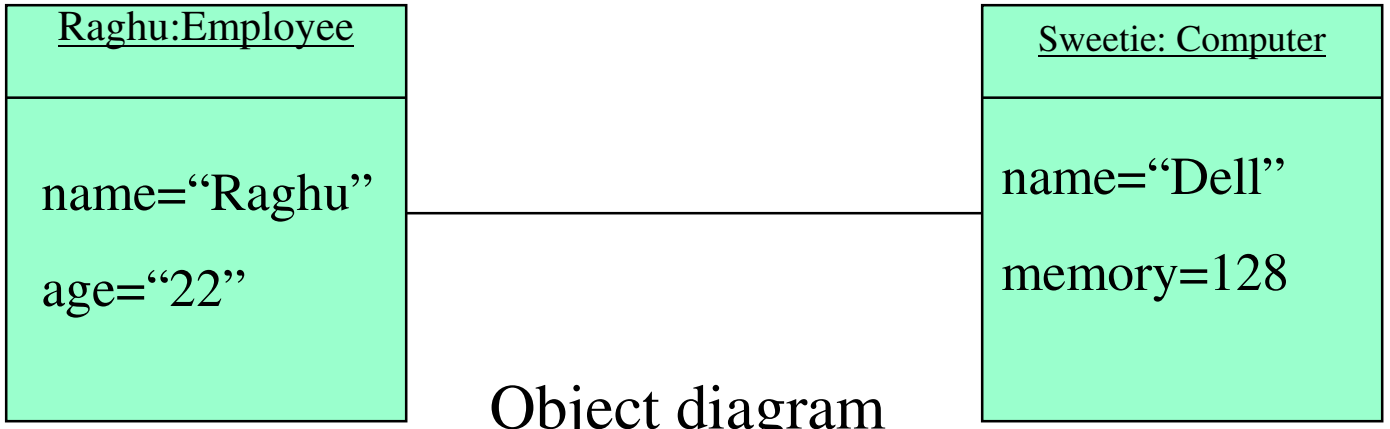
- Association
  - Unidirectional
  - Bi directional
- Generalization
- Aggregation
- Dependency
- Instantiation
- Refinement
- Realization



Associations

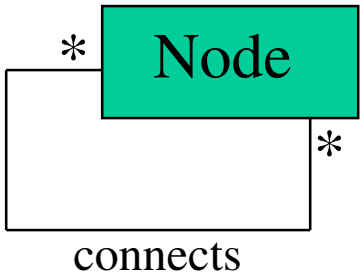


Class diagram

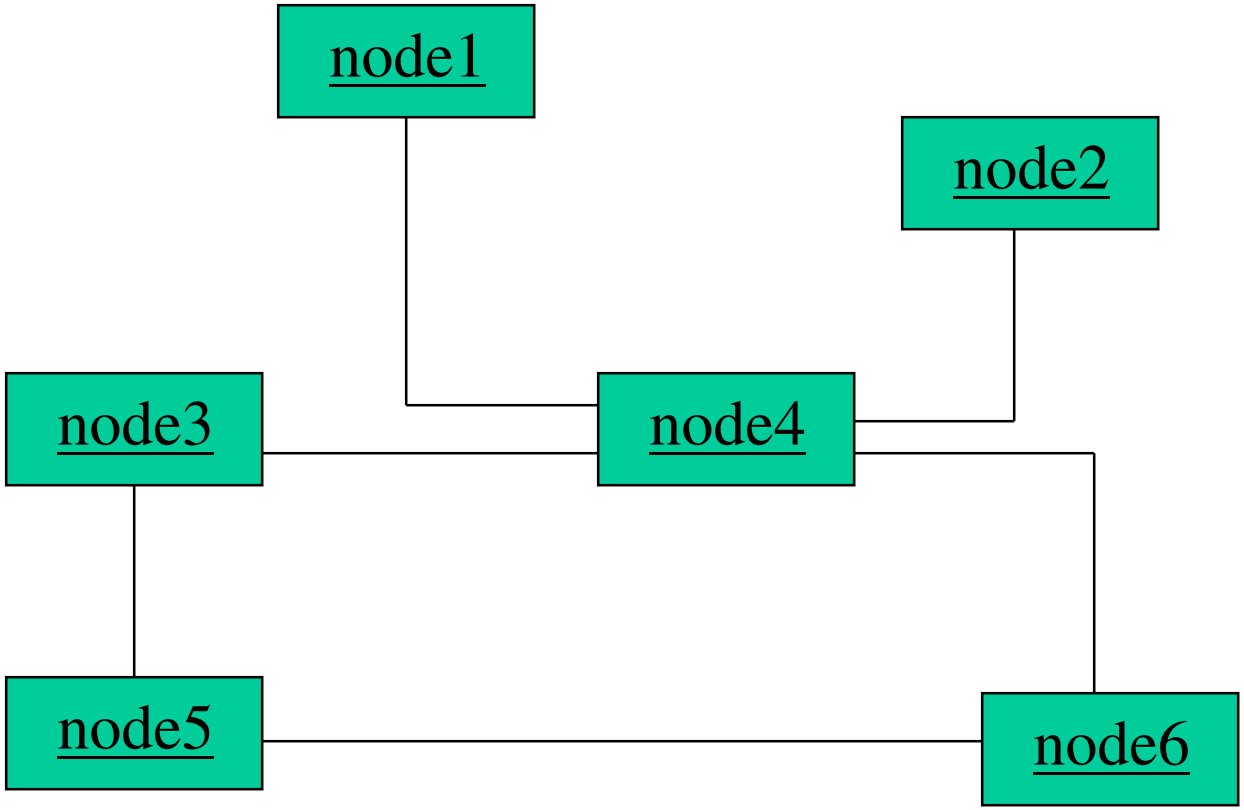


Object diagram

Recursive Association

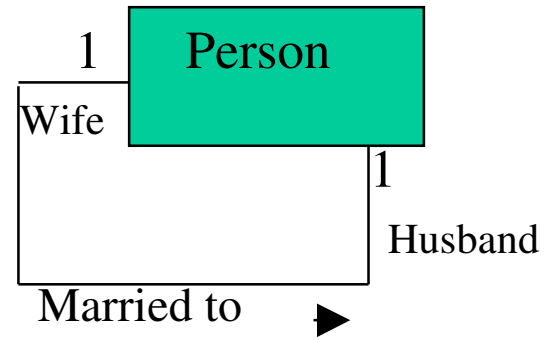


Class diagram



Object diagram

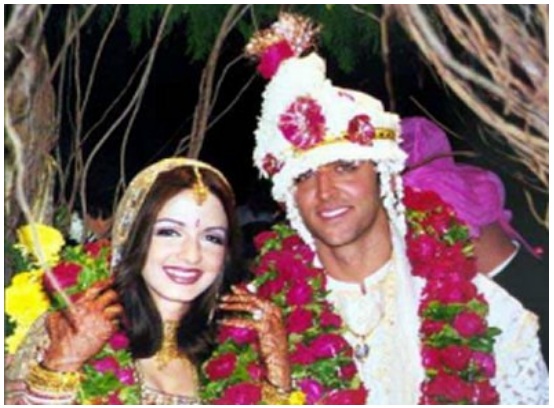
Roles



Ghauri Khan — Shahrukh Khan

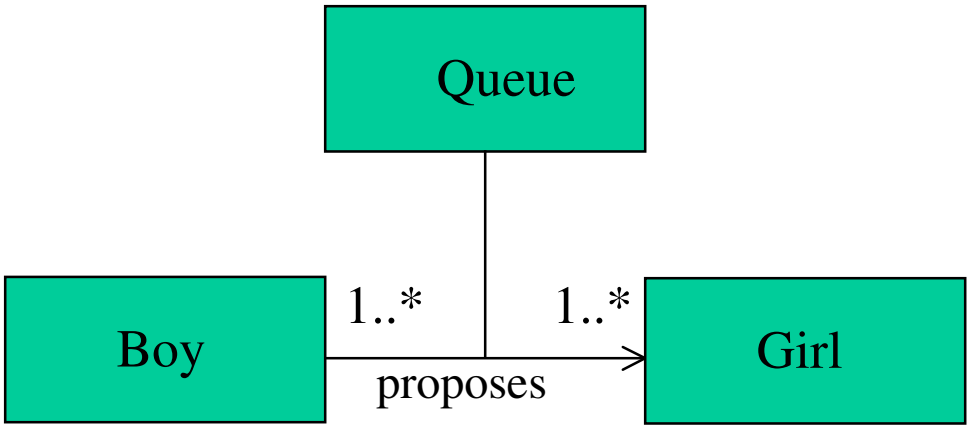
Suzzane Roshan — Hritik Roshan

Aish Rai — Raghu

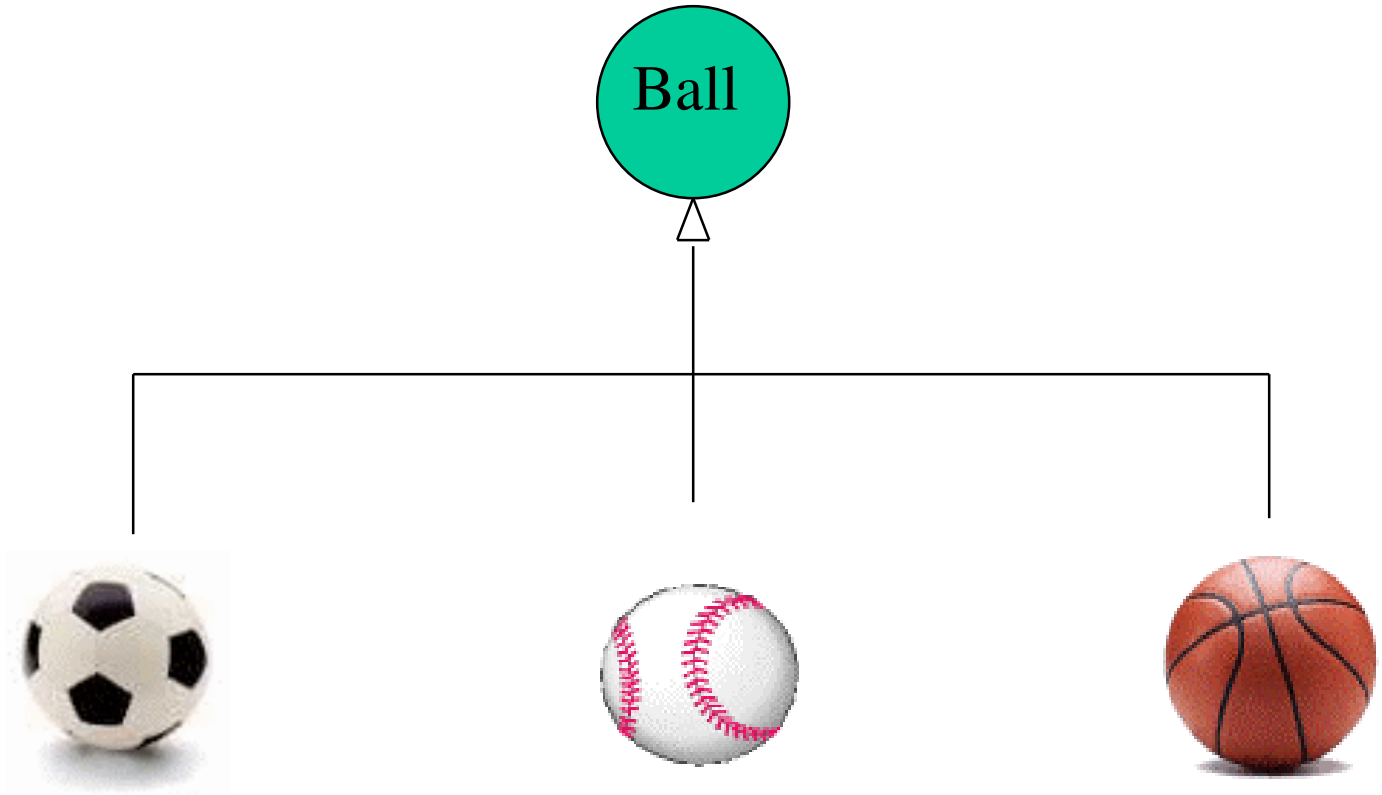


Association Class

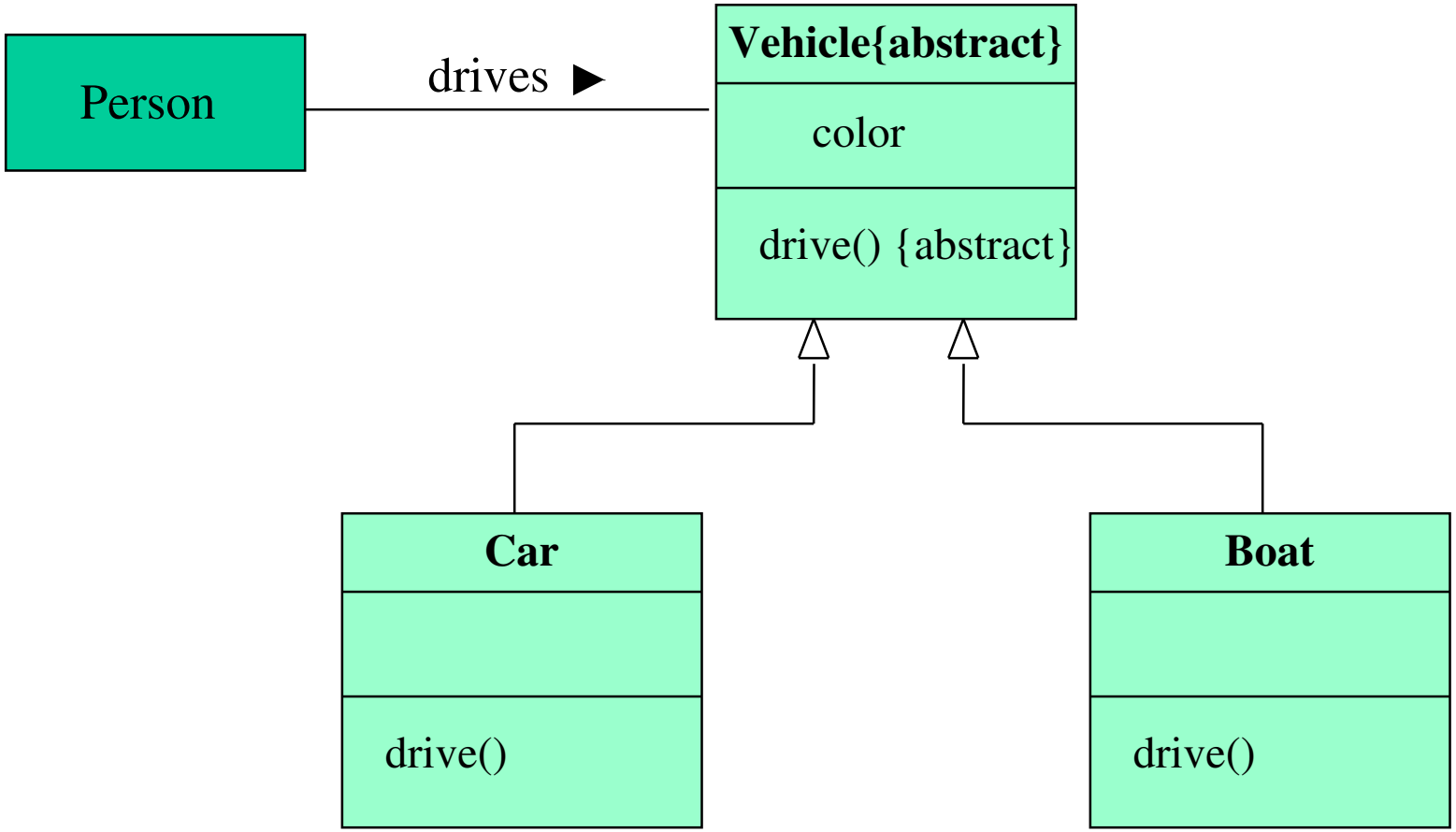
O, my Luve is like a red, red rose,
That's newly sprung in June.
O, my Luve is like the melodie,
That's sweetly played in tune.



Generalization



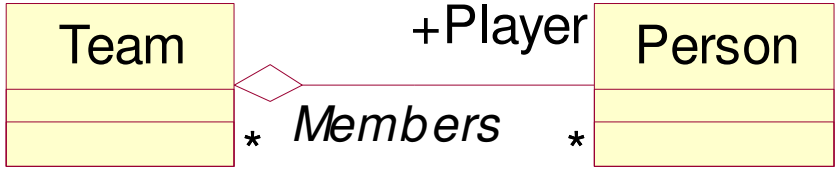
Generalization



Aggregation

- Aggregate

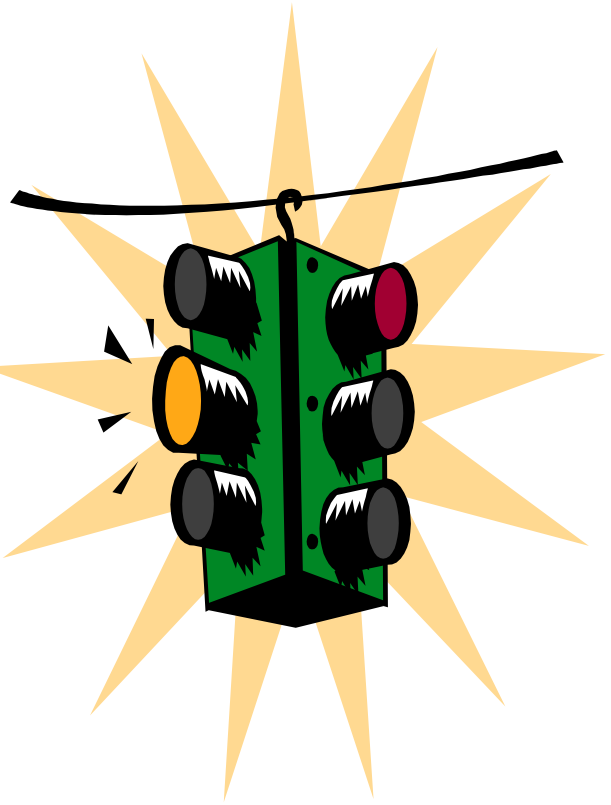
- Shared Aggregate



- Composition Aggregate

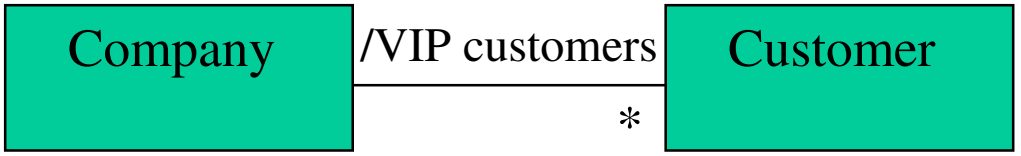
Constraints and Derivations(Rules)

- Rules
  - Constraints are for restriction
  - Derivations are for deducing
- Rules can be attached to any model
  - Attributes
  - Associations
  - Roles
  - Time constraints
- Rules are shown
  - in { } near the model element
  - in braces in a note symbol connected to the model element

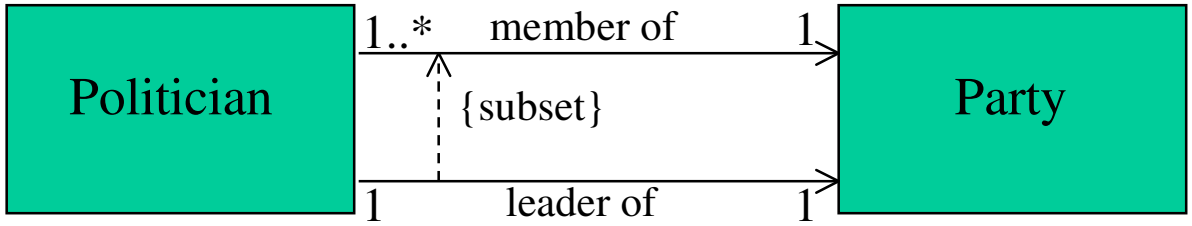


Associations can have rules

- Associations can be derived



- Associations can be constrained

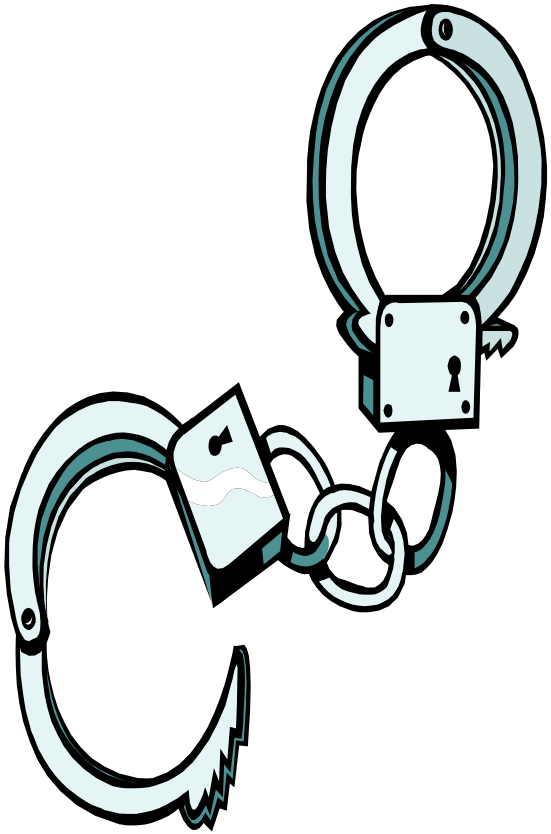


Attributes can have Rules

- Attributes can be derived
- Attributes can be constrained

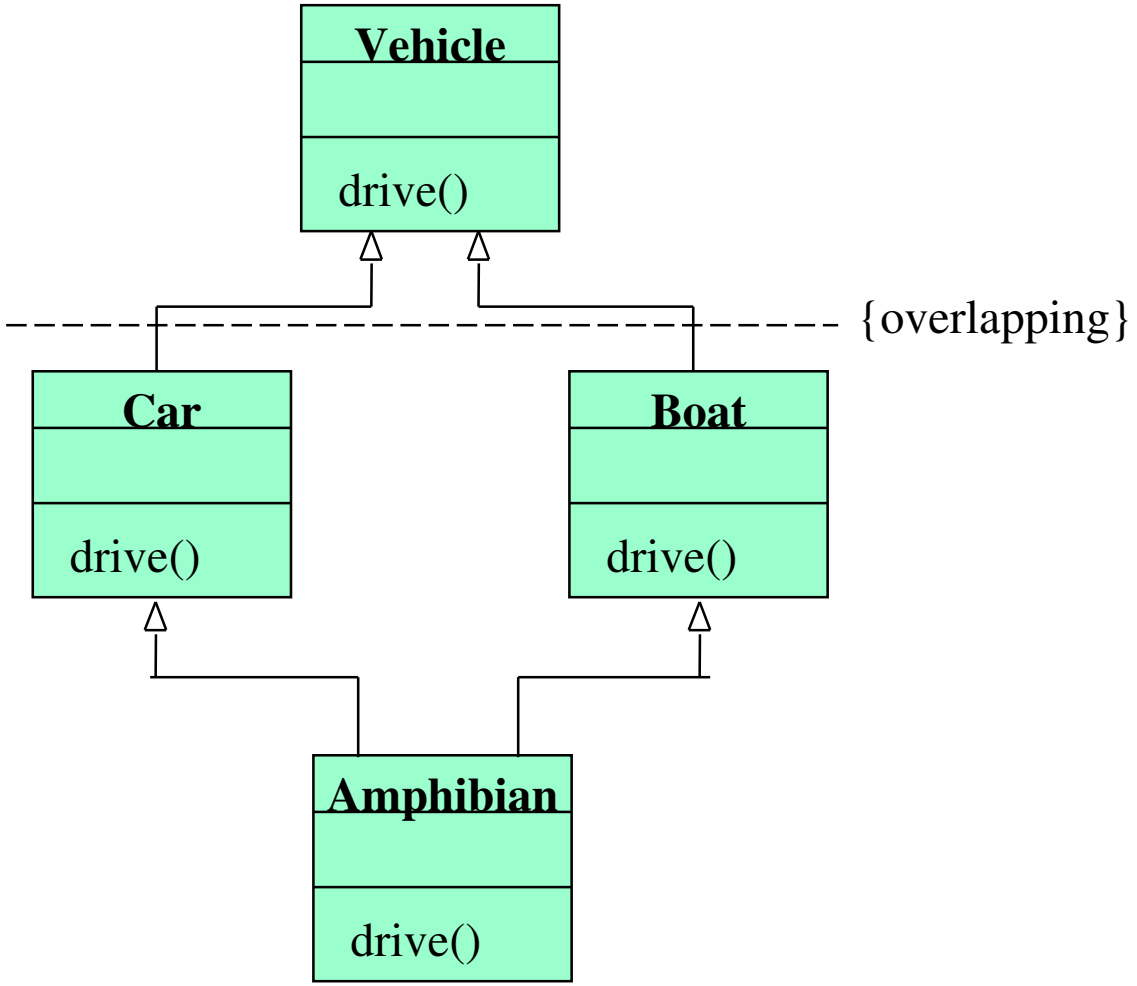
{Profit=SalesPrice-CostPrice}

Generalizations can have Constraints

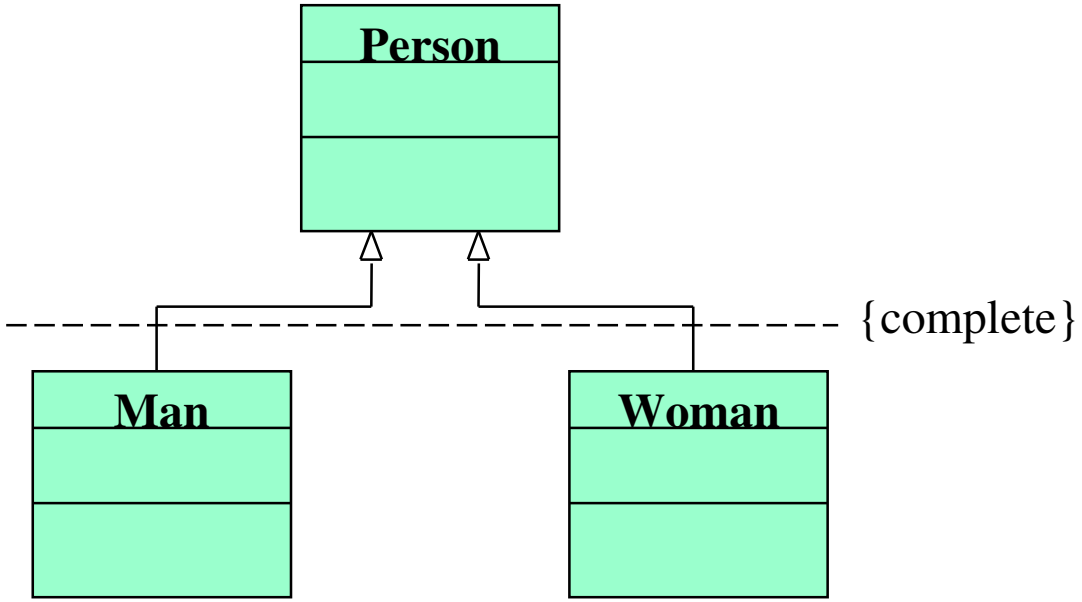


- Overlapping
- Disjoint
- Complete
- Incomplete

Overlapping Generalization



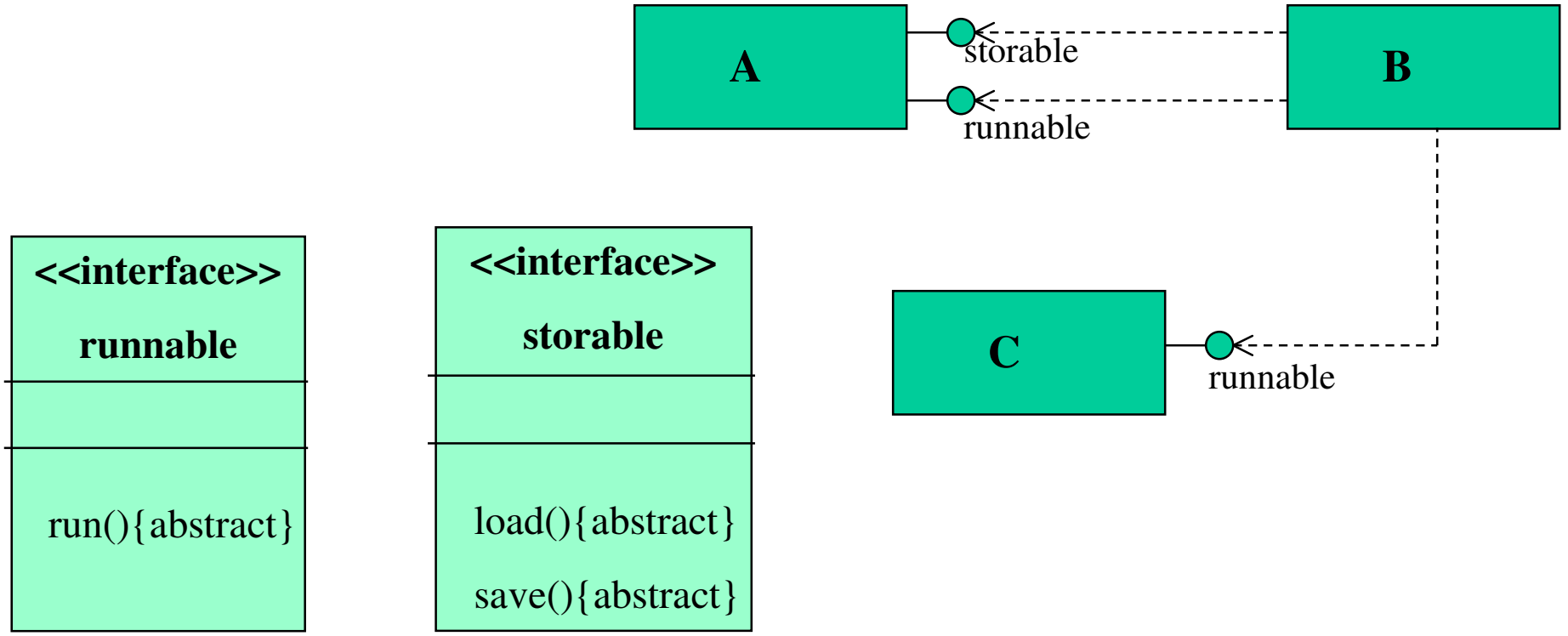
Complete Generalization



Interfaces

Interfaces

- What is an Interface?

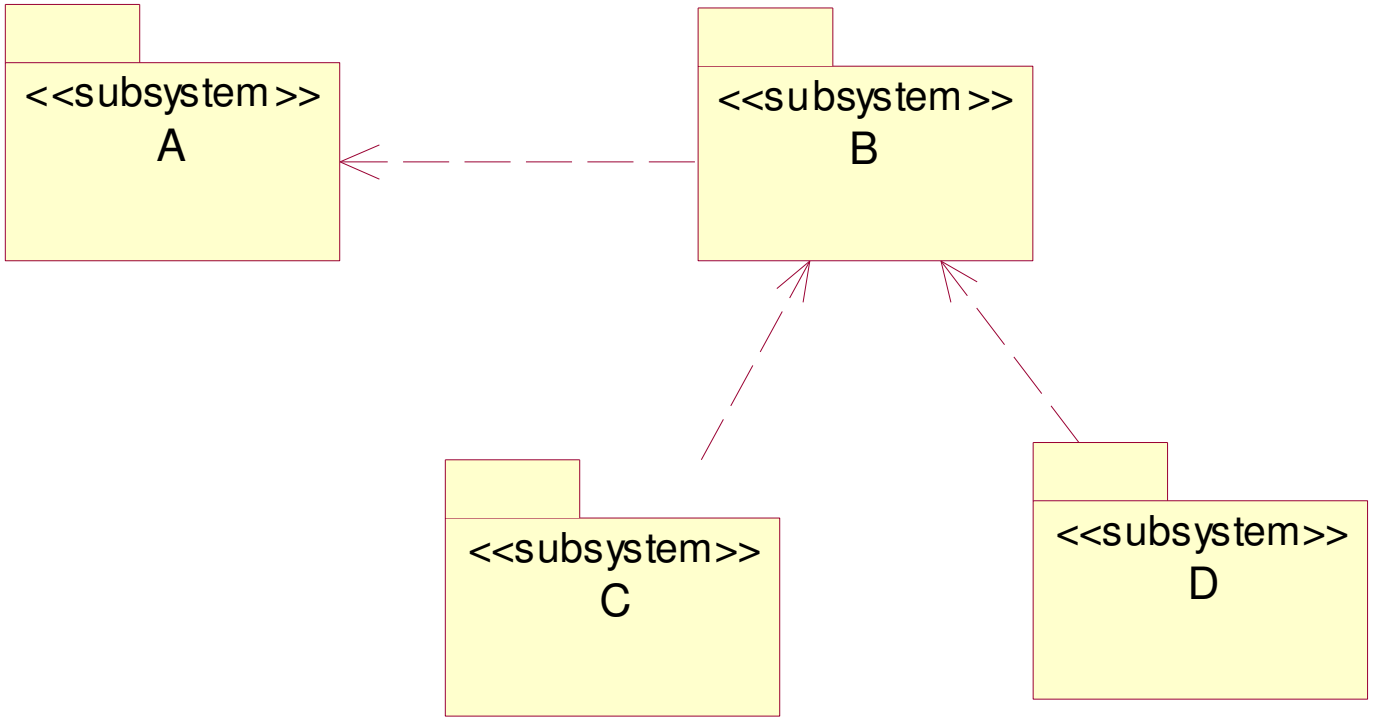


Packaging

Logical Packages

- Organize elements into semantically related groups
  - group classes, interfaces…
  - layer
  - subsystem
- Allowed relationships between packages
  - refinement
  - dependency
  - generalization
- Package has similarities to Aggregation
  - composed aggregation (if it owns the model elements)
  - shared aggregation (if it imports the model elements)

Dependency between Packages

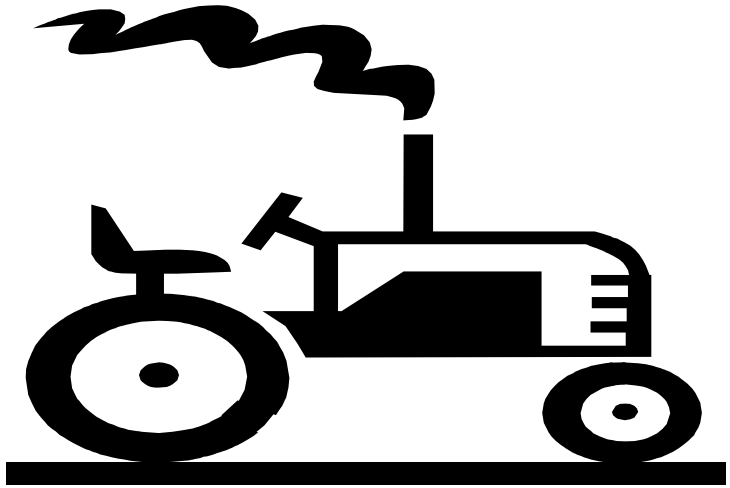


Coffee Break

Think about it:

Aggregation vs Generalization

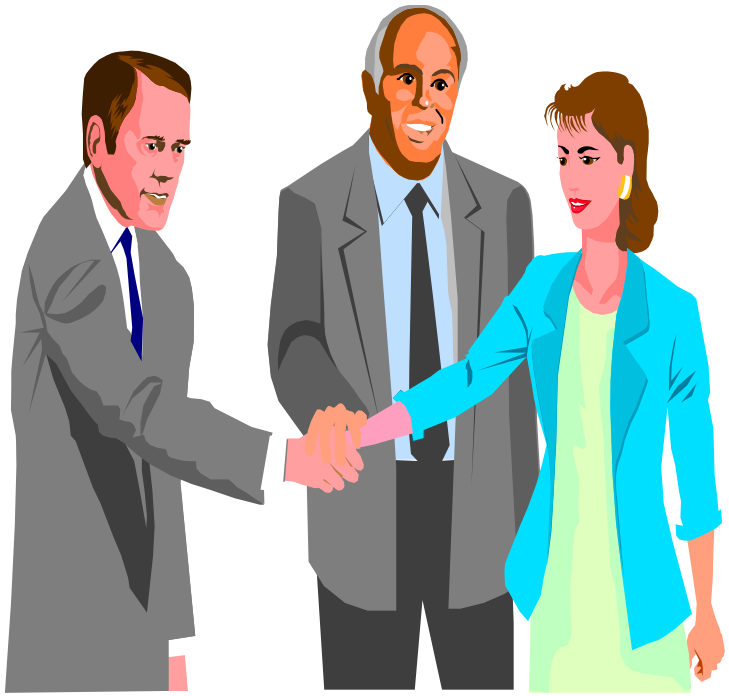
Dynamic Modeling

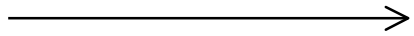
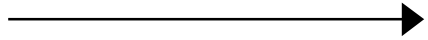
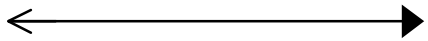


Dynamics

- State (focus on state)
- Sequence(focus on time)
- Collaboration(focus on space)
- Activity(focus on work)

Interactions



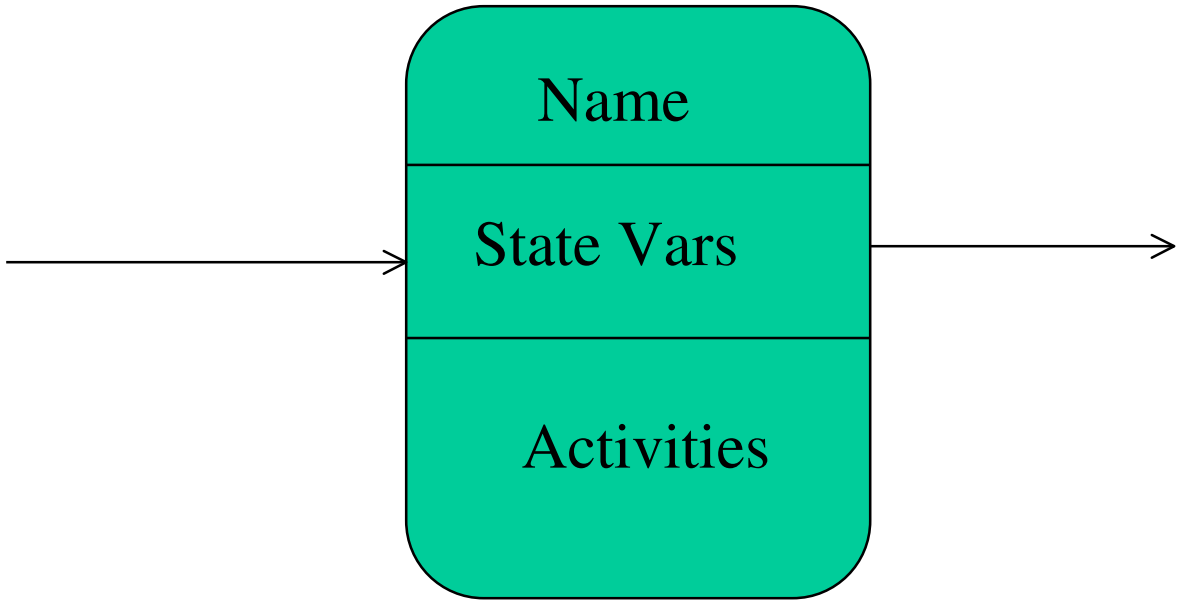
- Simple 
- Synchronous 
- Asynchronous
- Synchronous with (immediate) return 

State Diagrams

On/off

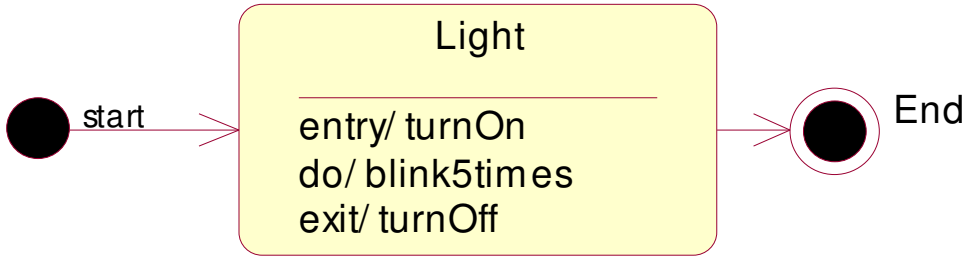
State Diagrams

- the sequences of states of an object
- the events causing a transition from one state to another
- the actions that result from a state or activity change



State

- Captures history
- Changes with or w/o an external event



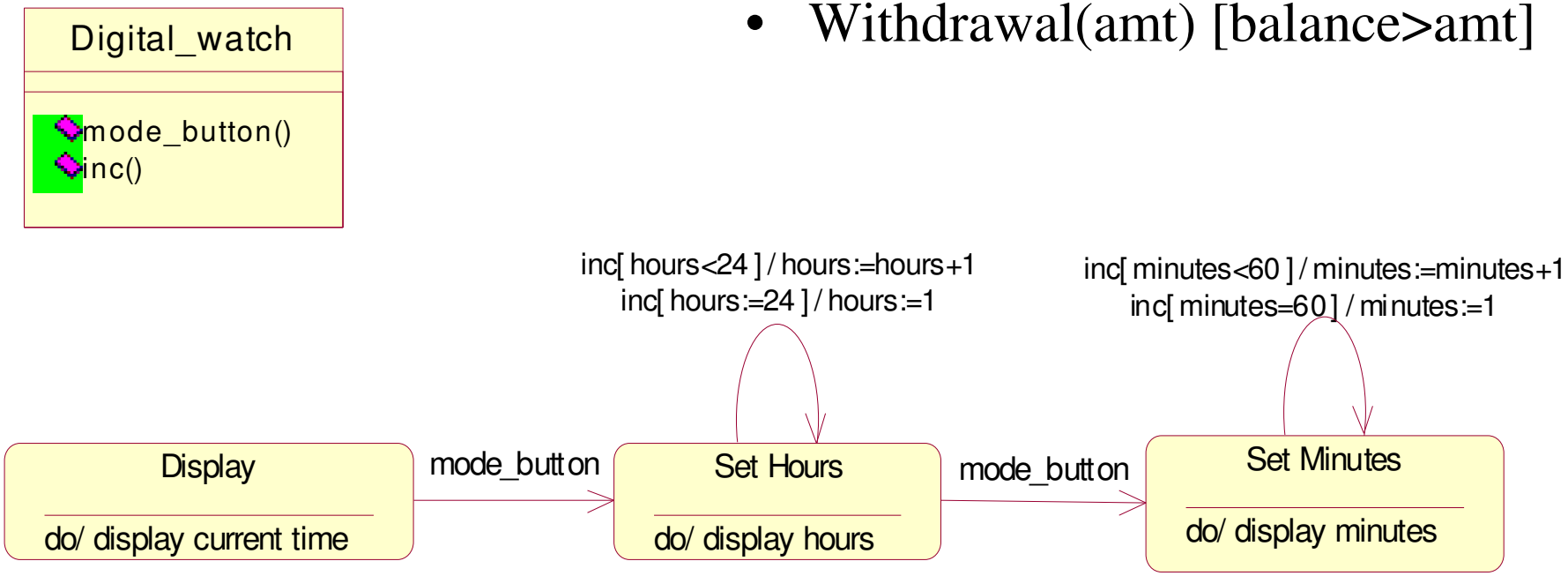
State transition

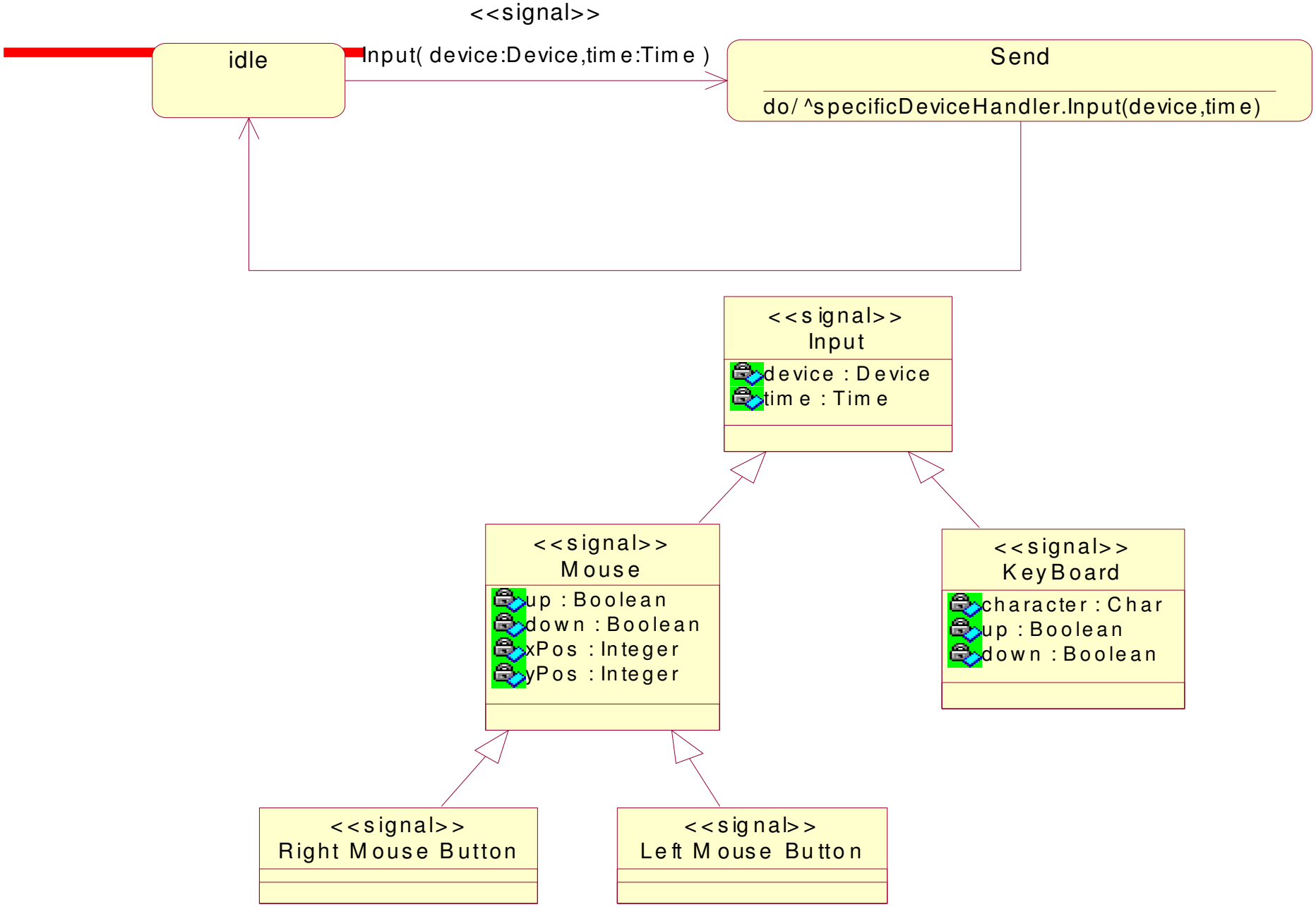
- event
- guard condition
- action
- send event
- send arguments
- send target

Event (args) [condition] / Action ^target.someEvent (args)

Transitions with a Guard condition

- [t=15 sec]
- [number of invoices > 15]
- Withdrawal(amt) [balance>amt]




<<signal>>
Input(device:Device,time:Time)
idle
Send
do/ ^specificDeviceHandler.Input(device,time)
<<signal>>
Input
device : Device
time : Time
<<signal>>
Mouse
up : Boolean
down : Boolean
xPos : Integer
yPos : Integer
<<signal>>
KeyBoard
character : Char
up : Boolean
down : Boolean
<<signal>>
Right Mouse Button
<<signal>>
Left Mouse Button

Sequence Diagrams

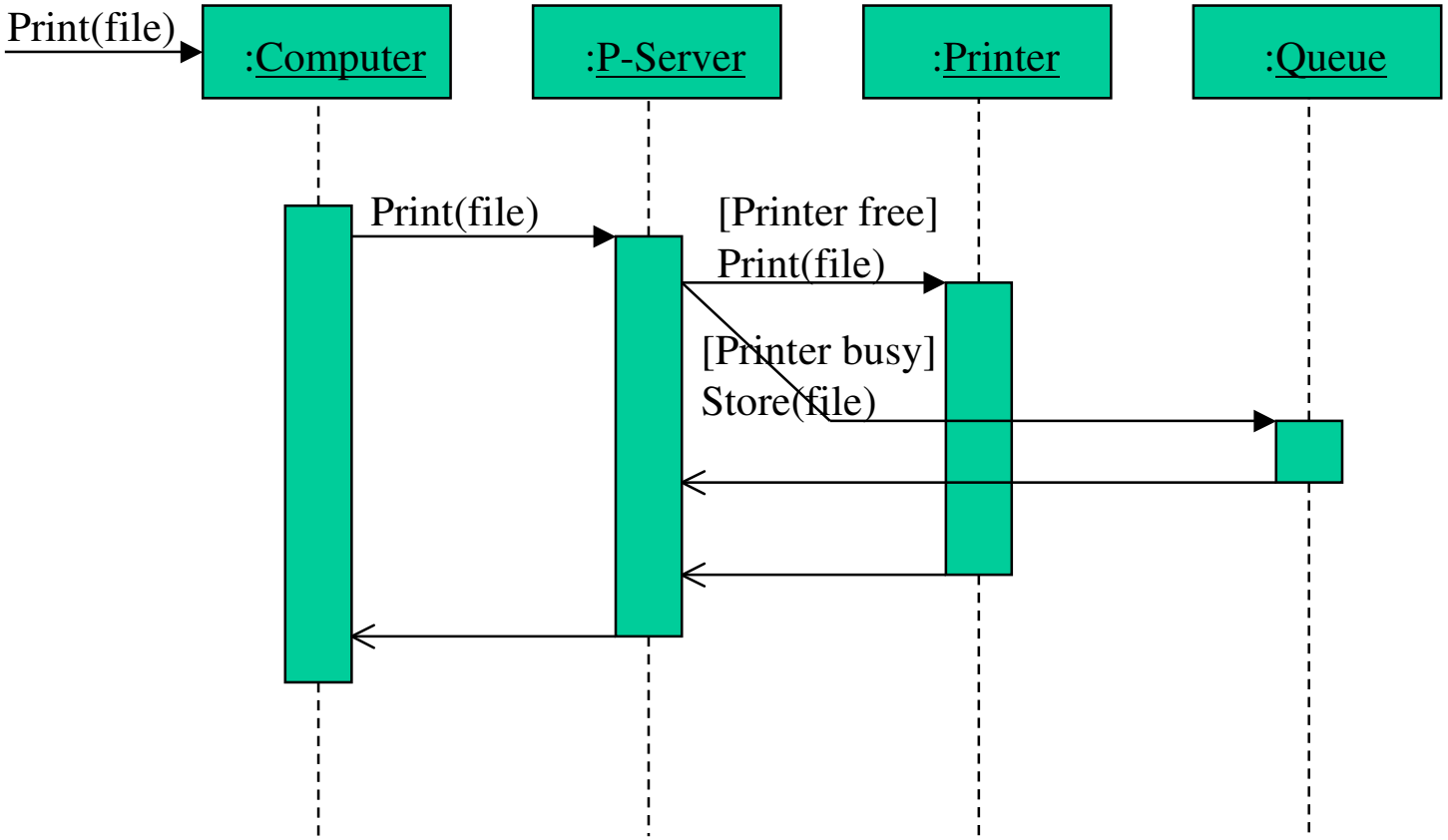
Sequence Diagrams

- The vertical axis shows time
- The horizontal axis shows objects
  - Objects represented as rectangles
  - Vertical dashed line indicates the object's life
- Two Forms
  - Generic form
    - More than one scenario → branches
  - Instance form
    - Only one scenario → no branches

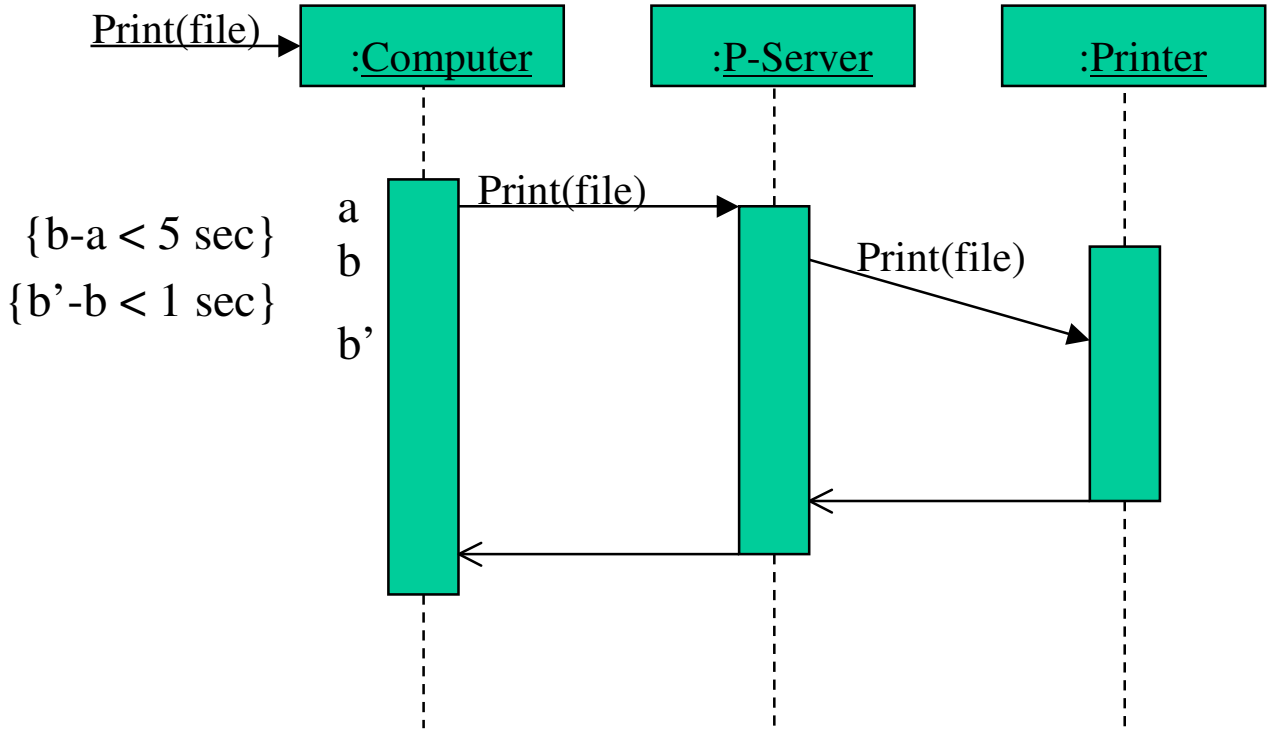
Sequence Diagrams

- Activated object
  - Is executing its own code
  - Waiting for return of another object
  - Drawn as a thin rectangle on the object's lifeline

Sample # 1

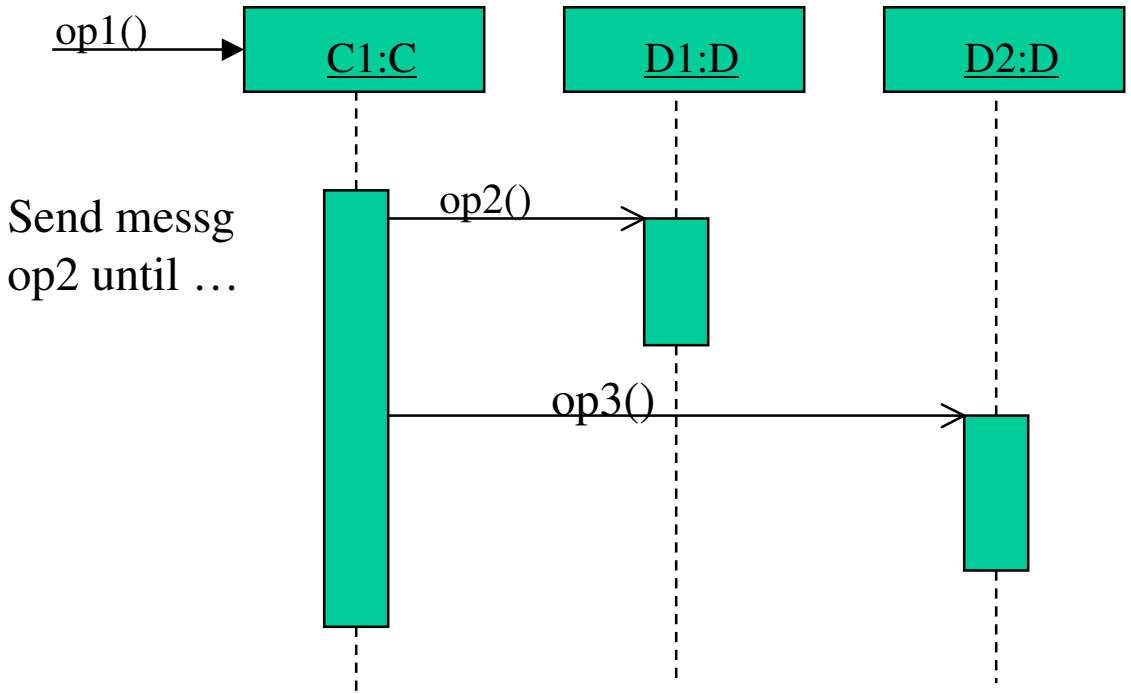

Print(file)
:Computer
:P-Server
:Printer
:Queue
Print(file)
[Printer free]
Print(file)
[Printer busy]
Store(file)

Sample # 2



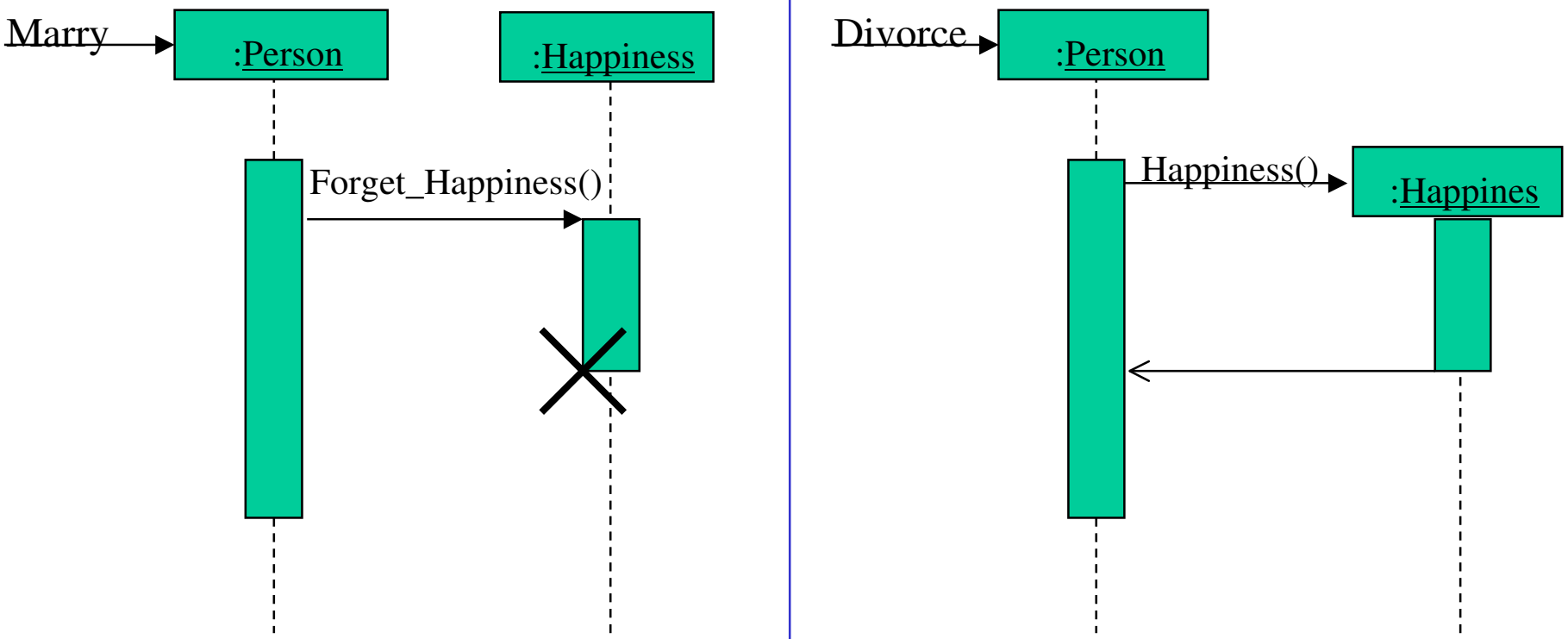
- Labels used to specify timing constraints
- A slanted arrow → substantial time between sending & receiving a message

Sample # 3



Iteration expressed in the margin

Sample # 4



Creation and Destruction of objects

Collaboration Diagrams

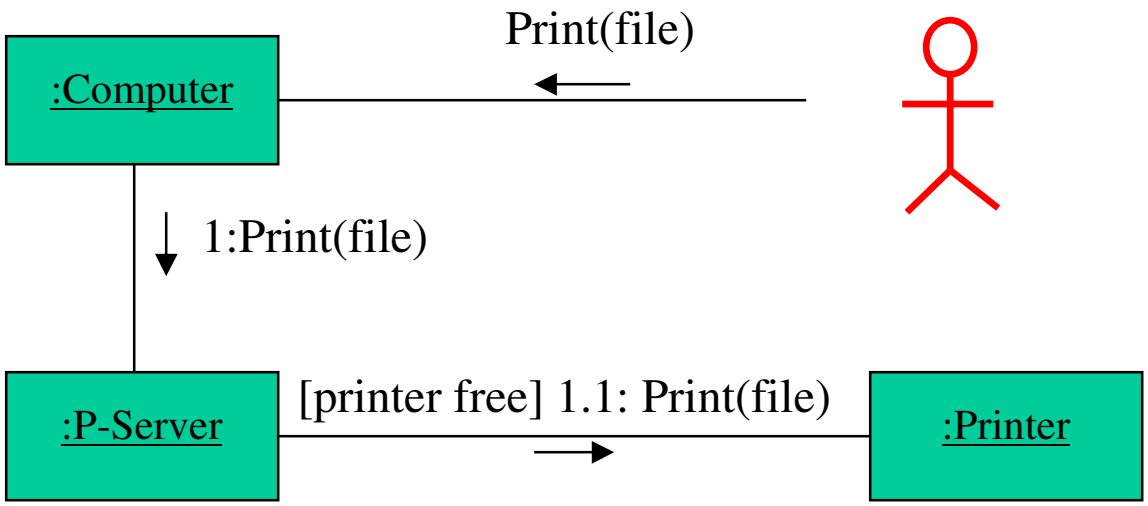
Collaboration Diagrams

- Focus on space
- Time sense
  - Message numbers
    - Simple numbering
    - Hierarchical numbering
      - Message 1 always starts a sequence of messages
      - Mssg 1.1 is the first nested mssg in handling mssg 1
      - Mssg 1.2 is the second nested mssg in handling mssg 1
      - An example sequence → 1,1.1,1.2,1.2.1,1.2.2,1.3
      - 1.2a and 1.2b are concurrent mssgs sent in parallel

Concepts

- Links
  - can apply ‘roles’
  - Supplier visibility
    - Field
    - Parameter
    - Local
    - global
  - Client visibility
- Lifetime of an object
  - {new}
  - {destroyed}
  - {transient}

Sample # 1


Print(file)
:Computer
1:Print(file)
:P-Server
[printer free] 1.1: Print(file)
:Printer

Activity Diagrams

Activity Diagrams

- State diagram + work done at each state
- Model elements
  - State
  - Activity
  - Start/end states
  - Transition
  - Synchronization elements
  - Decision box
  - Swimlanes

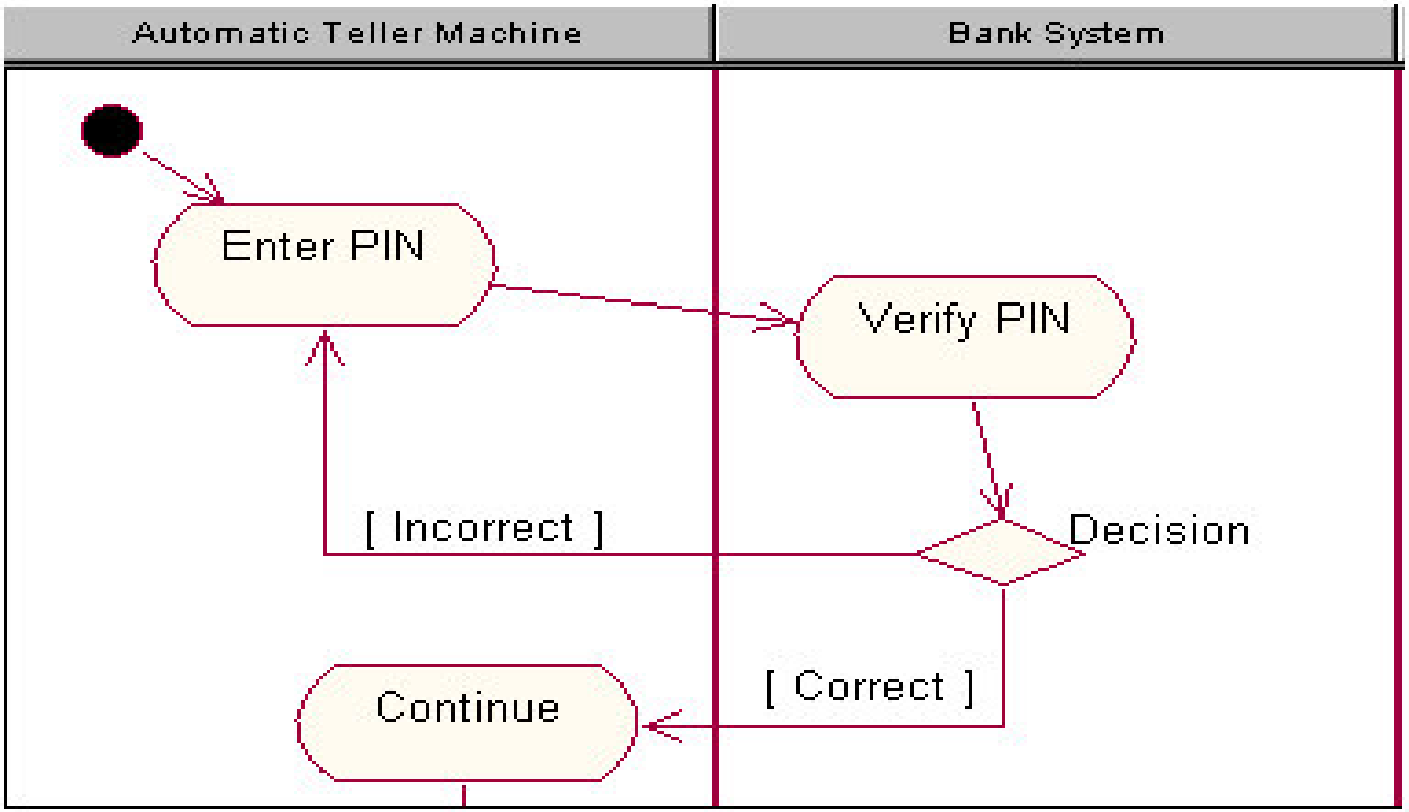
State and Activity Diagrams

- a special case of a state machine
  - most of the states are activities
  - most of the transitions are implicitly triggered

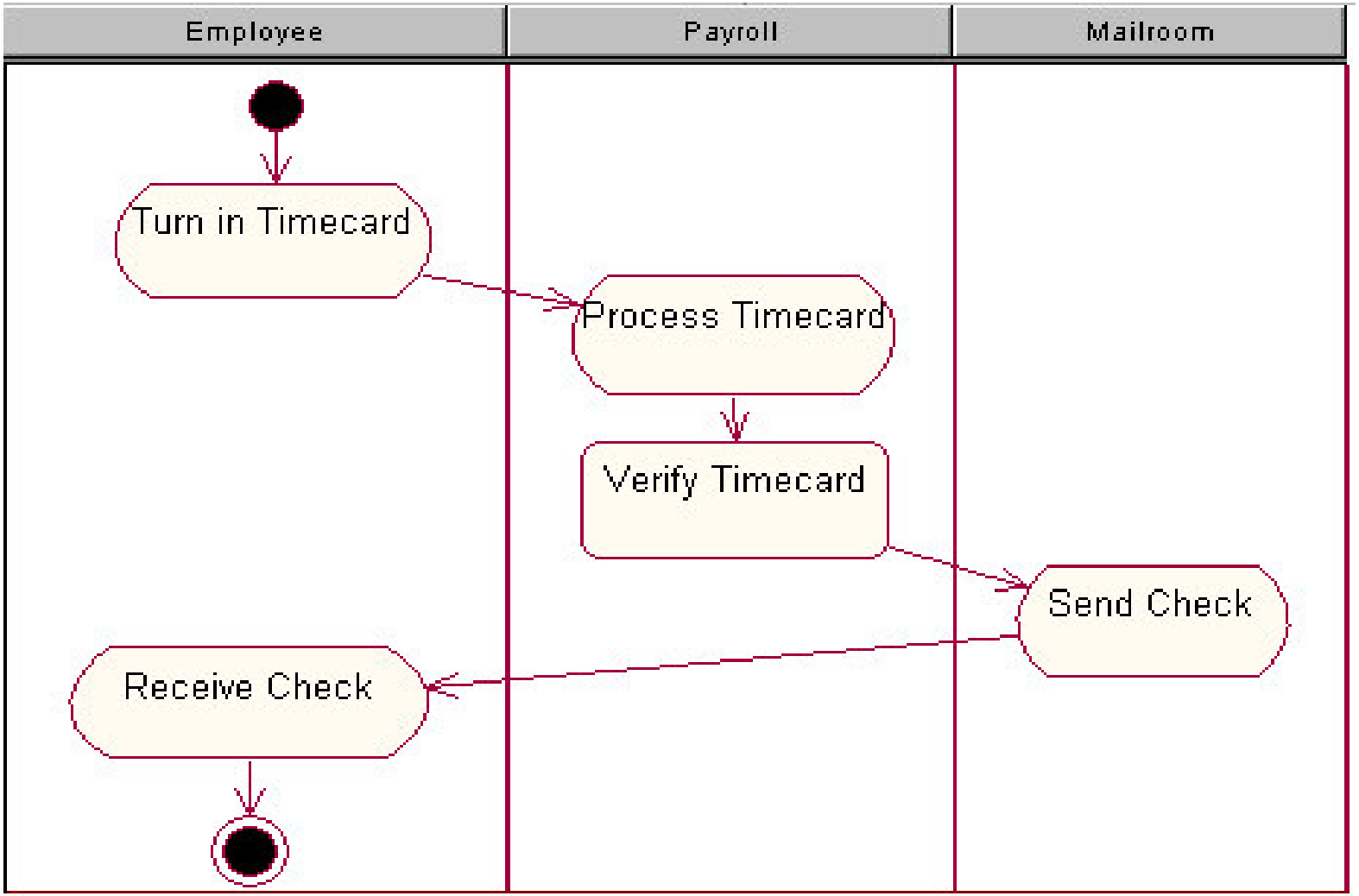
- Differences
  - activity diagrams are activity centric
  - statecharts are state centric

  - activity diagram typically model the sequence of activities in a process
  - statechart is better suited to model the discrete stages of an object's lifetime

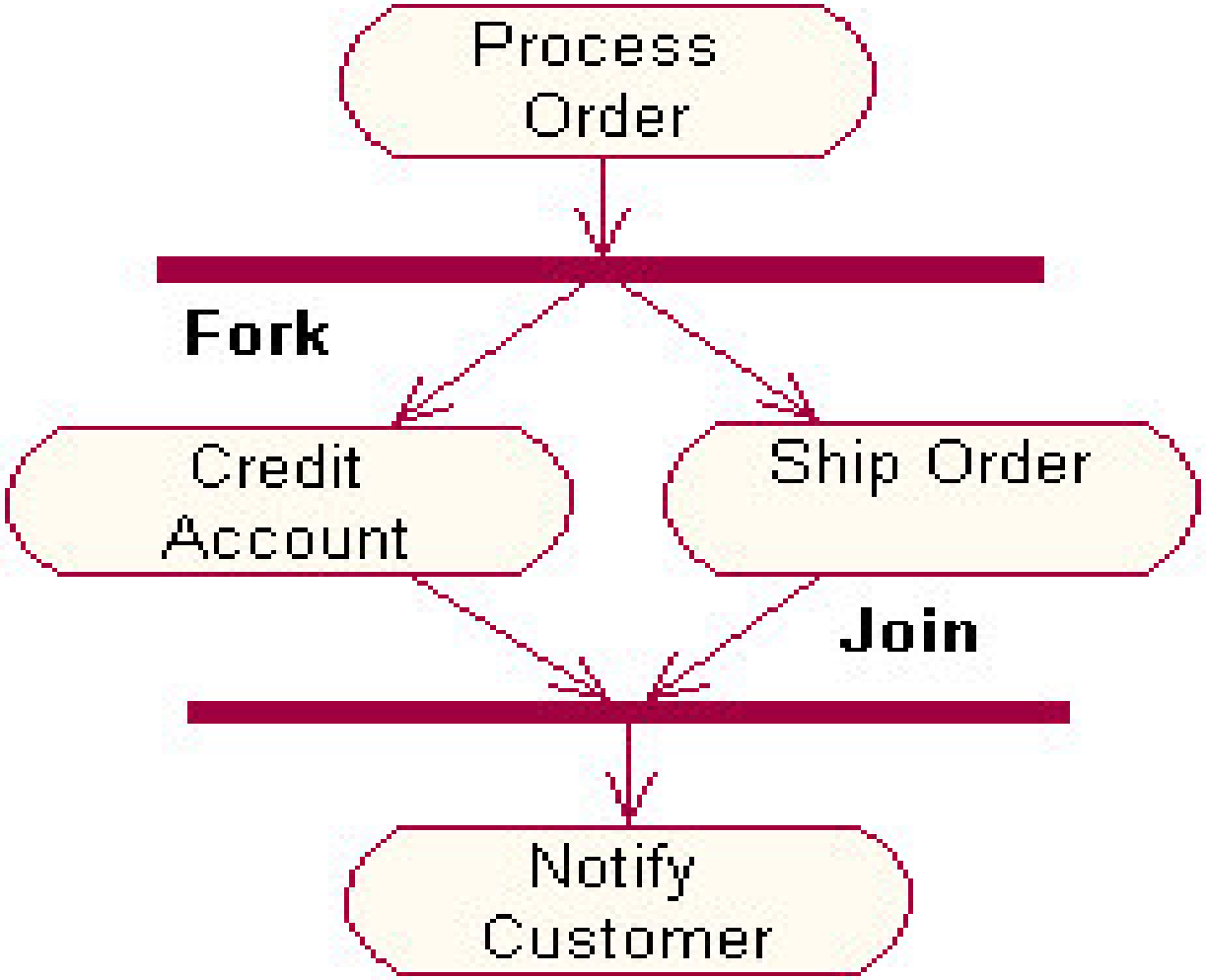
Sample # 1


Automatic Teller Machine
Bank System
Enter PIN
Verify PIN
[Incorrect]
Decision
[Correct]
Continue

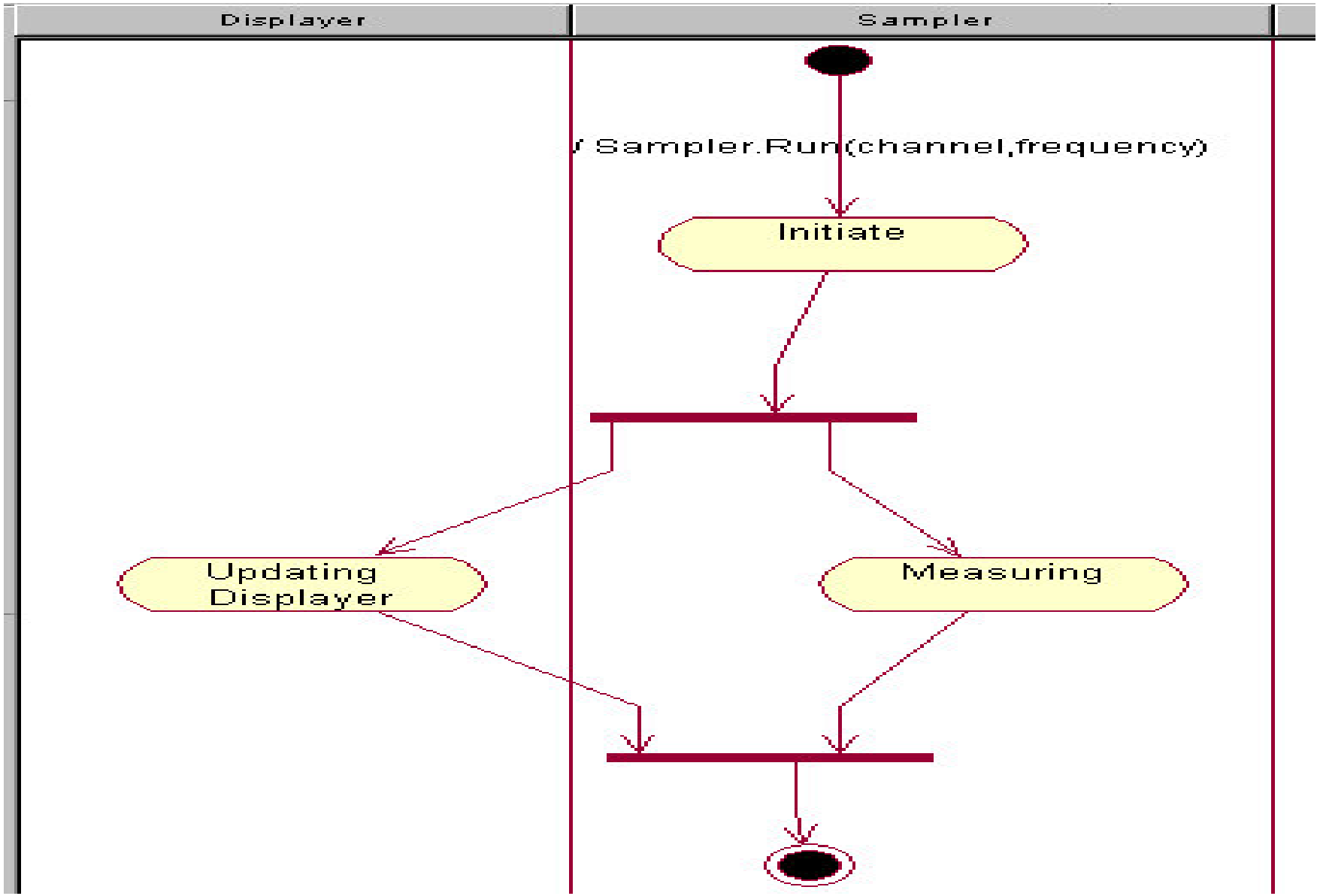
Sample #2


Employee
Payroll
Mailroom
Turn in Timecard
Process Timecard
Verify Timecard
Send Check
Receive Check

Sample #3


Process Order
Fork
Credit Account
Ship Order
Join
Notify Customer

Sample # 4


Displayer
Sampler
Sampler.Run(channel,frequency)
Initiate
Updating Displayer
Measuring

Physical Architecture

Physical architecture

- Objects in which files / processes ?
- On which computers do these programs/processes sit ?
- How are the computers connected ?
- Other hardware requirements
- Dependencies between different files
  - If a specific file is changed which other files need to recompiled

Component Diagrams

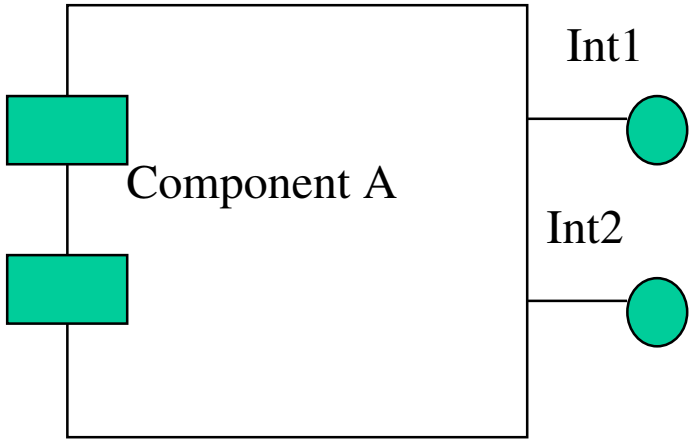
Component Diagrams

- Organization and dependencies among s/w components

- Component packages
- Components
- Interfaces
- Dependency relationships

Components

- Source code components
  - <<file>>
  - <<page>>
  - <<document>>
- Binary code components
  - <<library>>
- Executable components
  - <<application>>

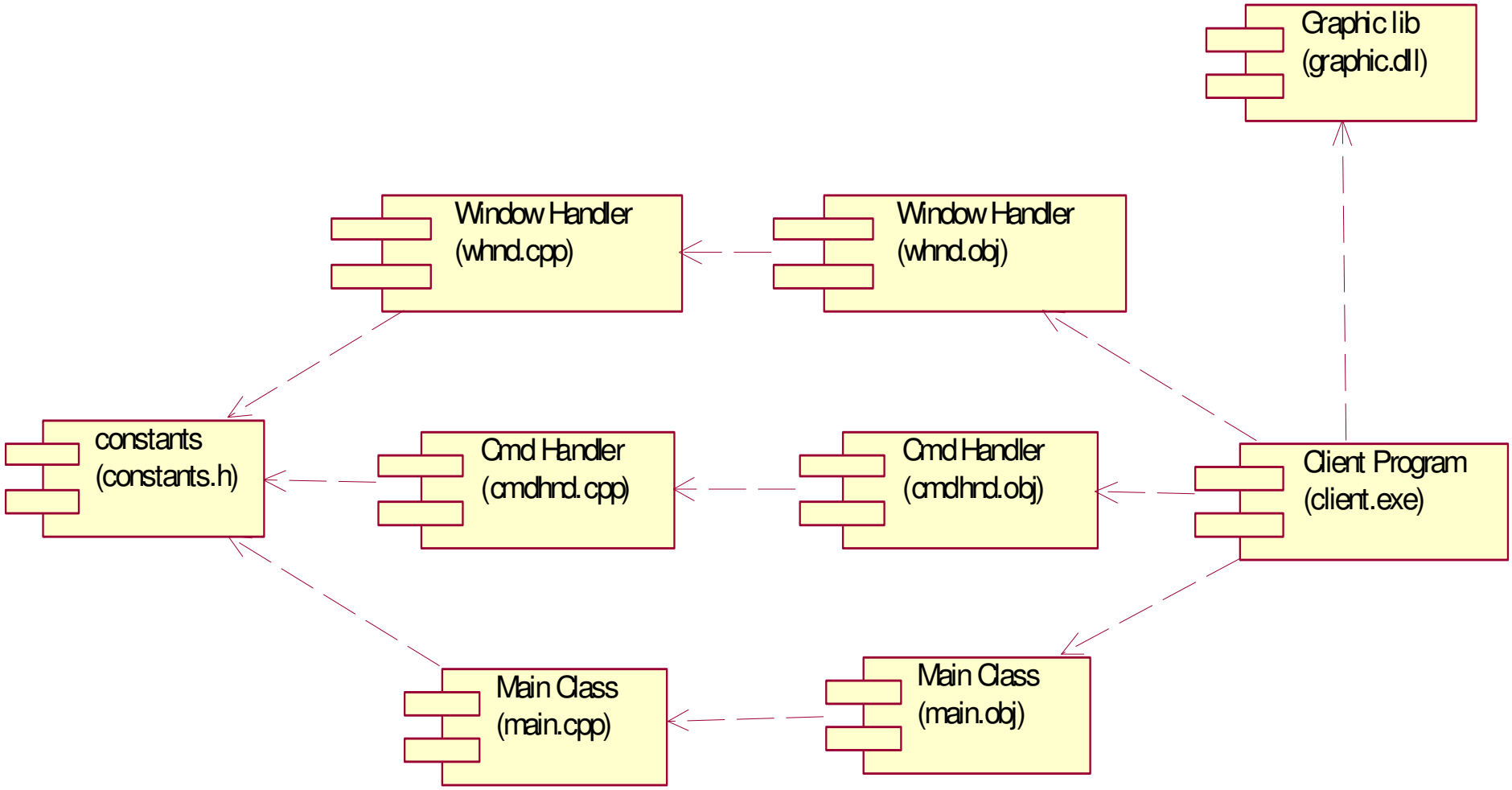


Rose representation

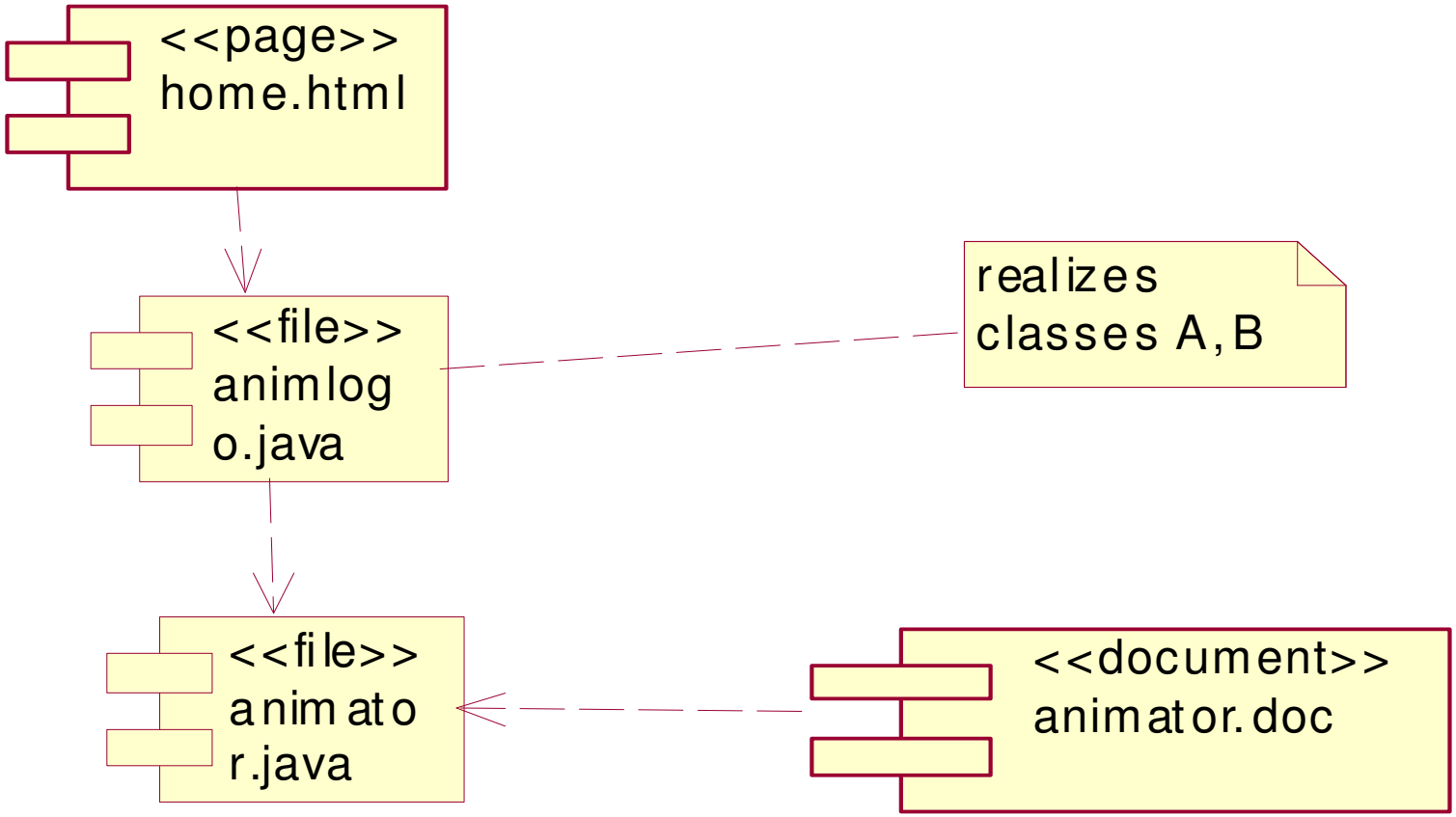
Component Packages

- Partition the physical model of the system
  - layer
  - subsystem
  - parallel the role of ‘logical packages for classes’ for physical entities
- Naming
  - usually file system directory name
- Relationships
  - Dependency
    - Component packages
    - Components
    - Interfaces

Sample # 1


Graphic lib
(graphic.dll)
Window Handler
(whnd.cpp)
Window Handler
(whnd.obj)
constants
(constants.h)
Cmd Handler
(cmdhnd.cpp)
Cmd Handler
(cmdhnd.obj)
Client Program
(client.exe)
Main Class
(main.cpp)
Main Class
(main.obj)

Sample # 2

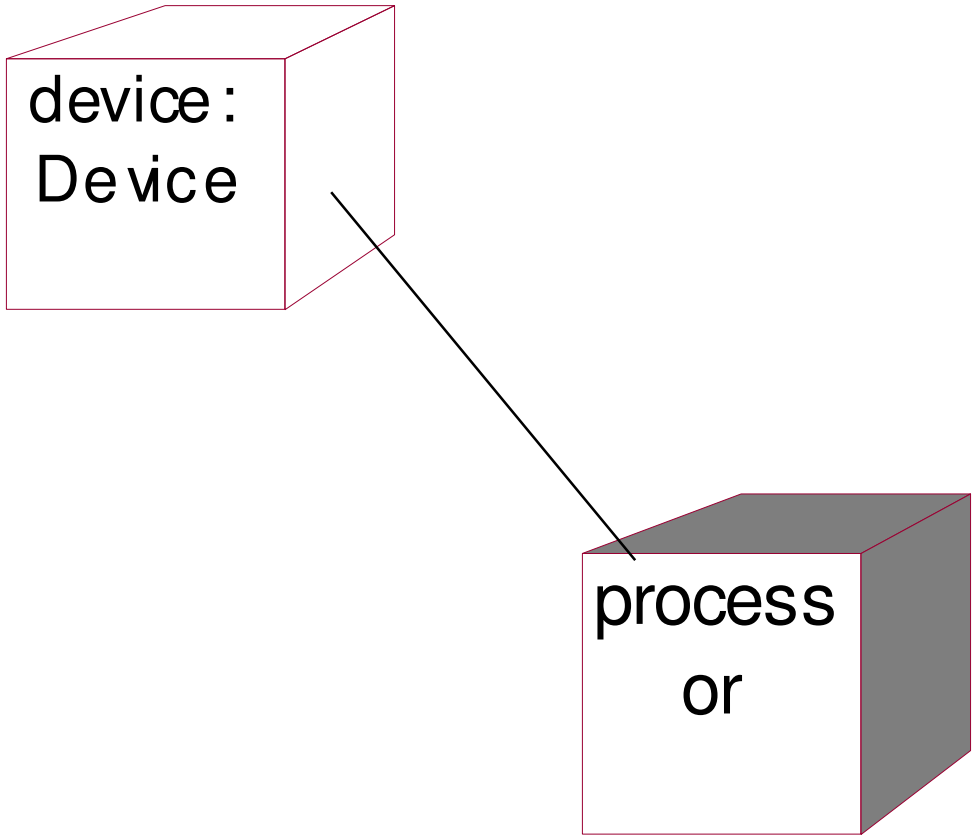


Dependencies between source-code components.

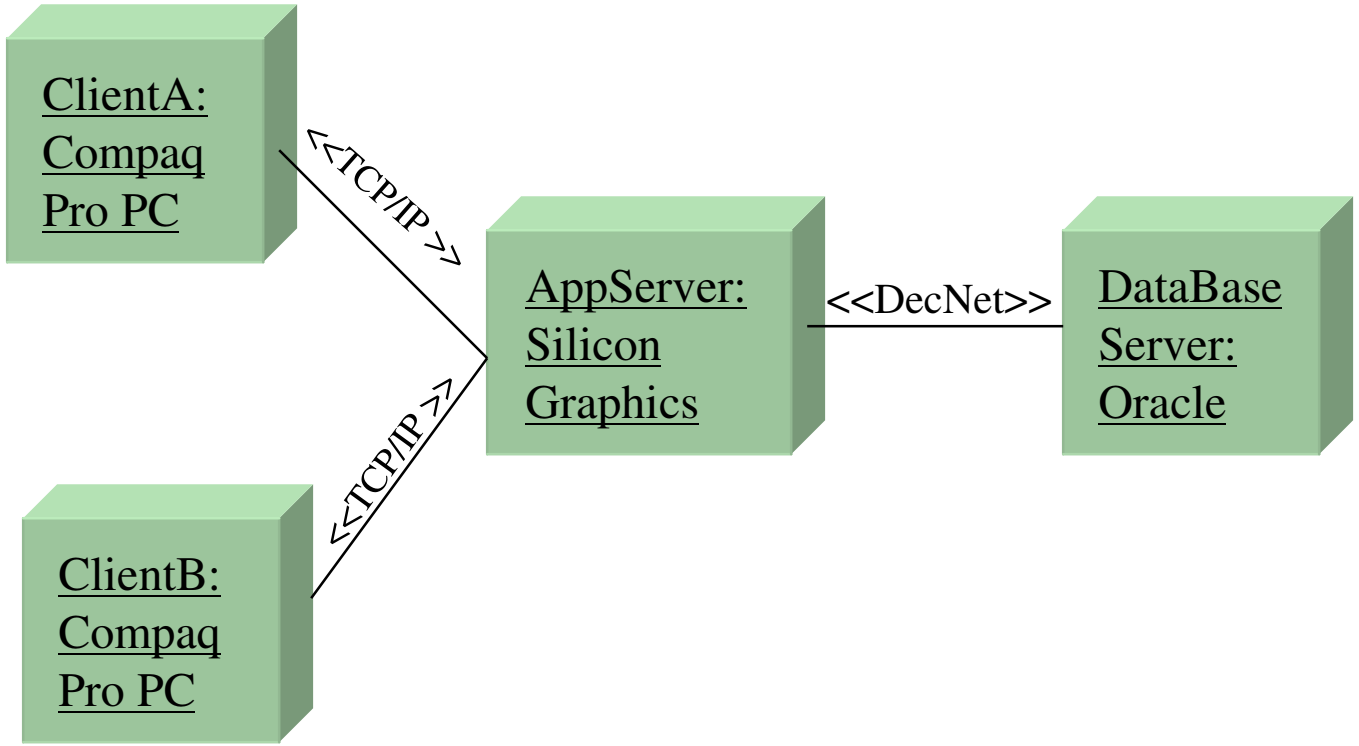
Deployment Diagrams

Deployment Diagrams

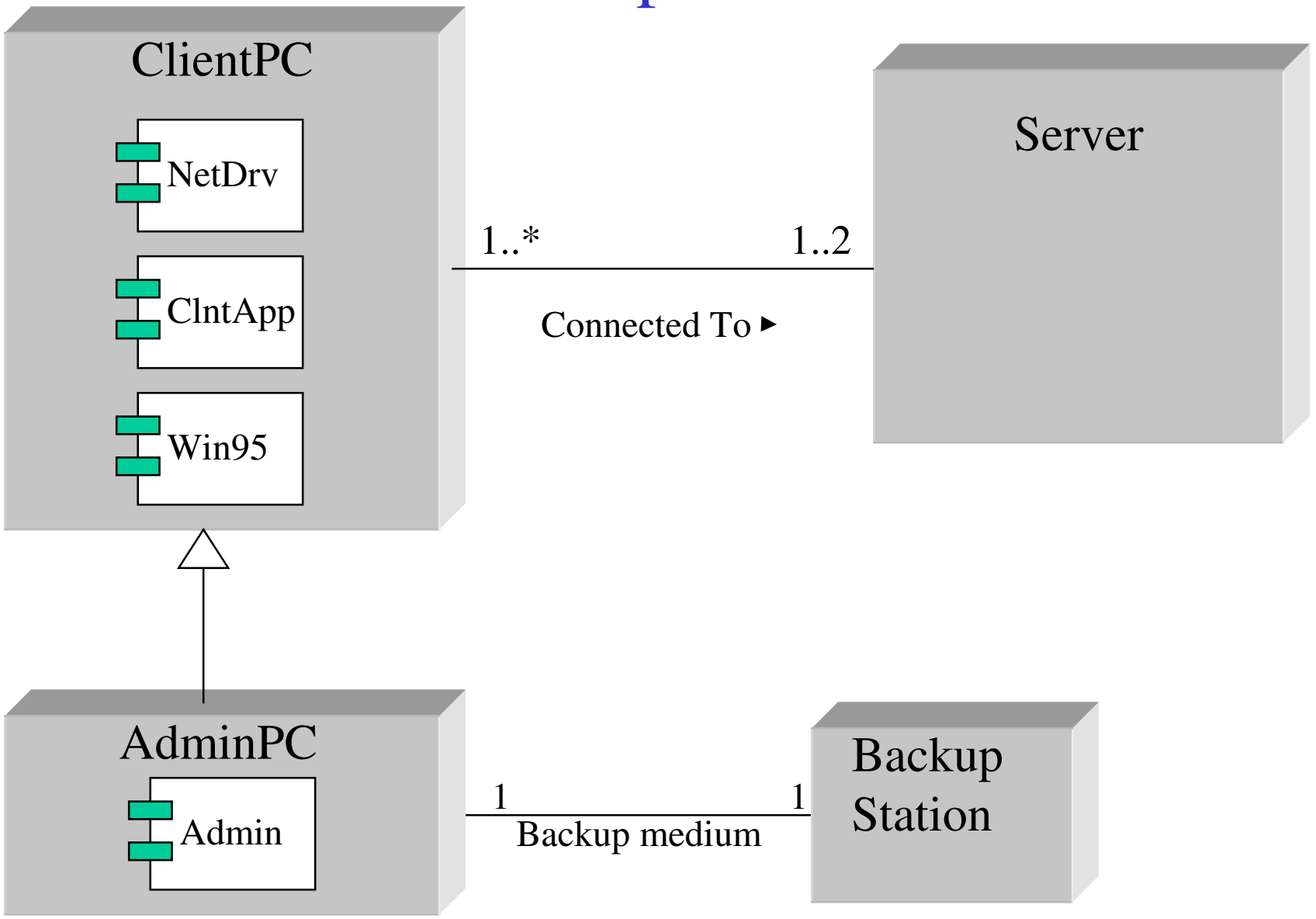
- Describes the runtime architecture and system topology
- Modeling elements
  - Processors
  - Devices
  - Connections

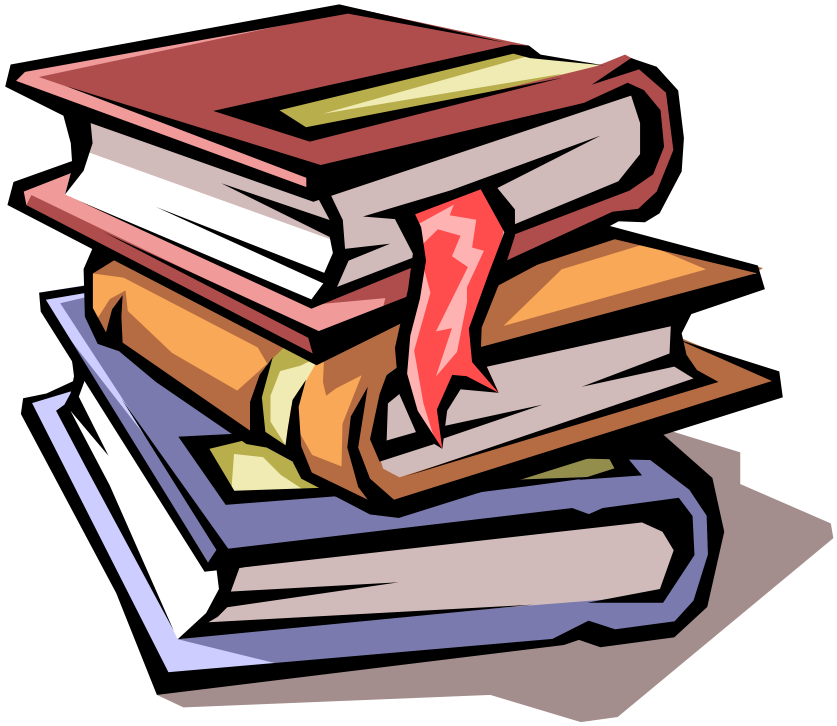


Sample # 1


ClientA:
Compaq
Pro PC
<<TCP/IP>>
AppServer:
Silicon
Graphics
<<DecNet>>
DataBase
Server:
Oracle
<<TCP/IP>>
ClientB:
Compaq
Pro PC

Sample #2


ClientPC
NetDrv
ClntApp
Win95
Server
1..*
1..2
Connected To ▸
AdminPC
Admin
1
1
Backup medium
Backup Station



Case Study

Wake UP !!You Win Chocolates for this !!! ☺

Library Application

- It is a support system for a library.
- A library lends books and magazines to borrowers, who are registered in the system, as are the books and magazines.
- A library handles the purchase of new titles for the library. Popular titles are bought in multiple copies. Old books and magazines are removed when they are out of date or in poor condition.
- The librarian is an employee of the library who interacts with the customers(borrowers) and whose work is supported by the system.
- A borrower can reserve a book or magazine that is not currently available in the library, so that when it is returned or purchased by the library, that person is notified.The reservation is canceled when the borrower checks out the book or through an explicit canceling procedure.
- The library can easily create,update, and delete information about the titles, borrowers,loans and reservations in the system.
- The system can run on all popular technical environments (UNIX , Windows, OS/2, etc.) and has a modern GUI.
- The system is easy to extend with new functionality.