CIS 285: UML Models

Sequence Diagrams

Sequence Diagrams

- a representation of behavior as a series of sequential steps over time
- used to depict work flow, message passing and how elements in general cooperate over time to achieve a result
- each sequence element is arranged in a horizontal sequence, with messages passing back and forward between elements
- elements (actors, objects, classes, etc.) have a dashed stem called a lifeline, where that element exists and potentially takes part in the interactions

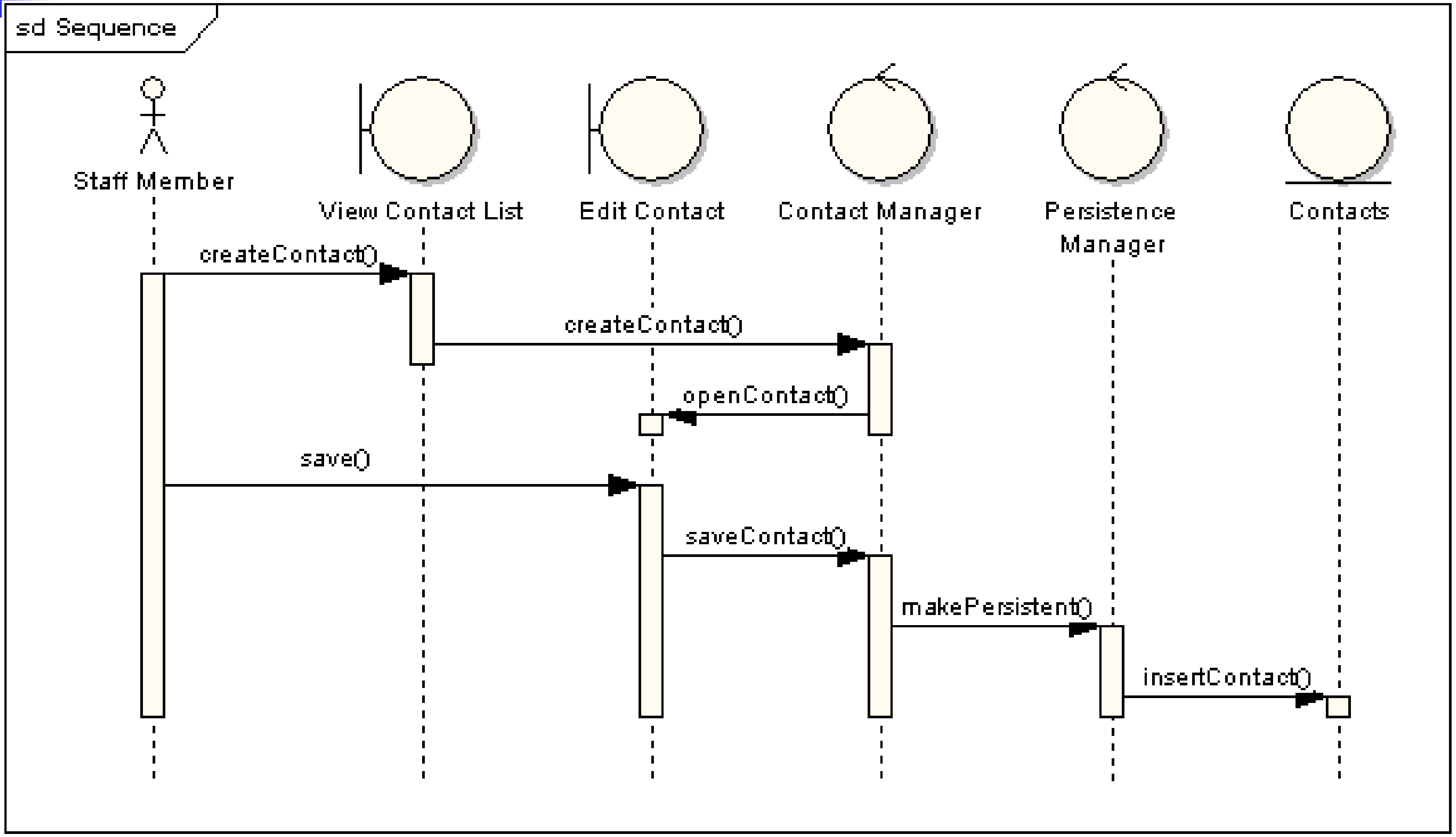
Sequence Diagrams …

- a form of interaction diagram which shows objects as lifelines running down the page, with their interactions over time represented as messages drawn as arrows from the source lifeline to the target lifeline
- they are good at showing which objects communicate with which other objects; and what messages trigger those communications
- they are not intended for showing complex procedural and algorithmic logic

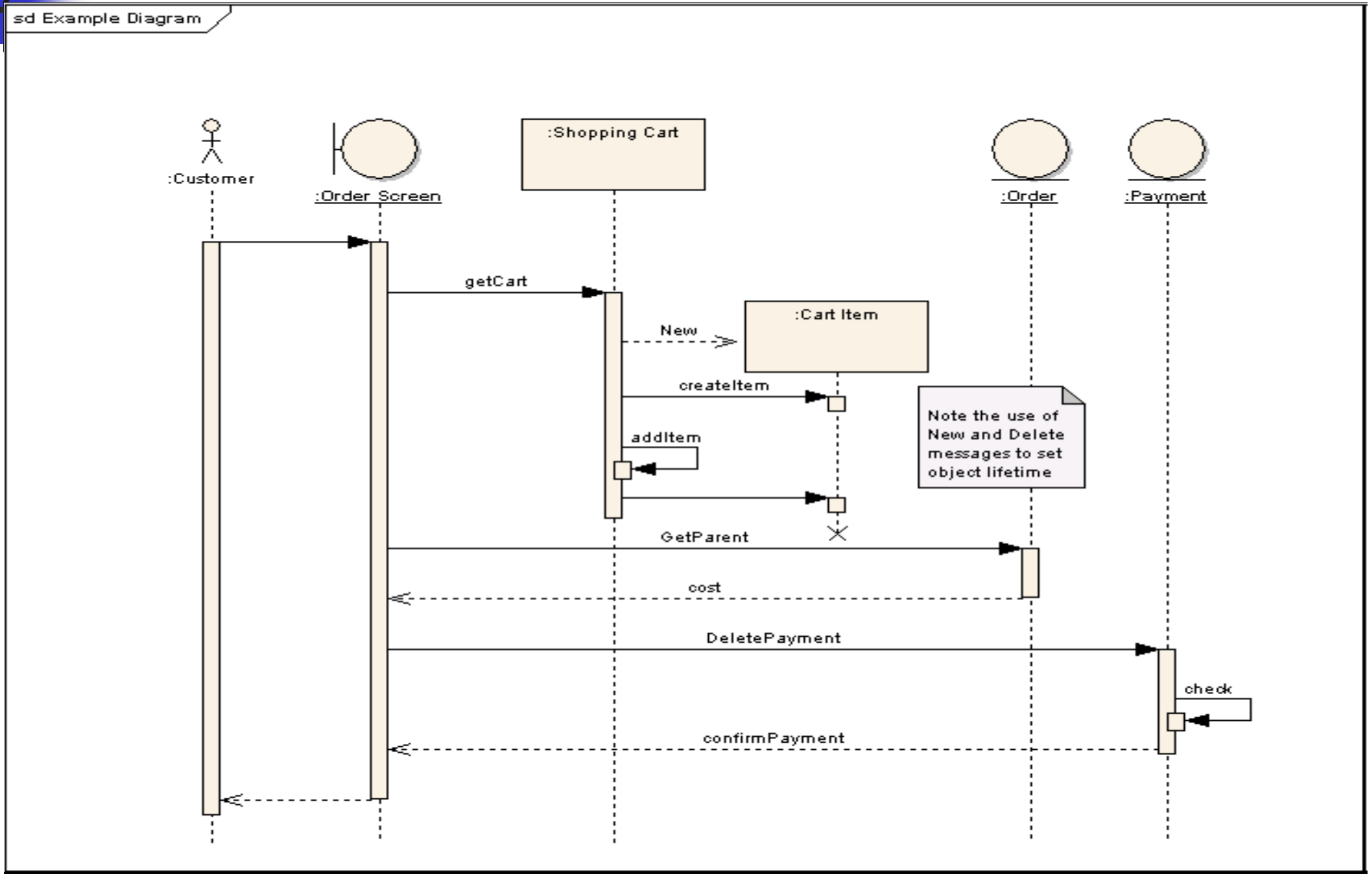
Sequence Diagram: Elements

- actor
- boundary
- control
- entity
- lifelines
- messages
- execution occurrence
- self messages
- lost and found messages
- lifeline start and end
- duration and time constraints
- combined fragments
- gate
- part decomposition
- state invariants / continuations

Sequence Diagram: Example 1



Sequence Diagram: Example 2



Object Types

- Entity Objects
  - Represent the persistent information tracked by the system (Application domain objects, “Business objects”)
- Boundary Objects
  - Represent the interaction between the user and the system
- Control Objects:
  - Represent the control tasks performed by the system

Sequence Diagrams: Boundary

- a stereotyped class that models some system boundary - typically a user interface screen
- used in the conceptual phase to capture users interacting with the system at a screen level (or some other boundary interface type)
- often used in sequence, communication, and analysis diagrams
- it is the View in the Model-View-Controller pattern.

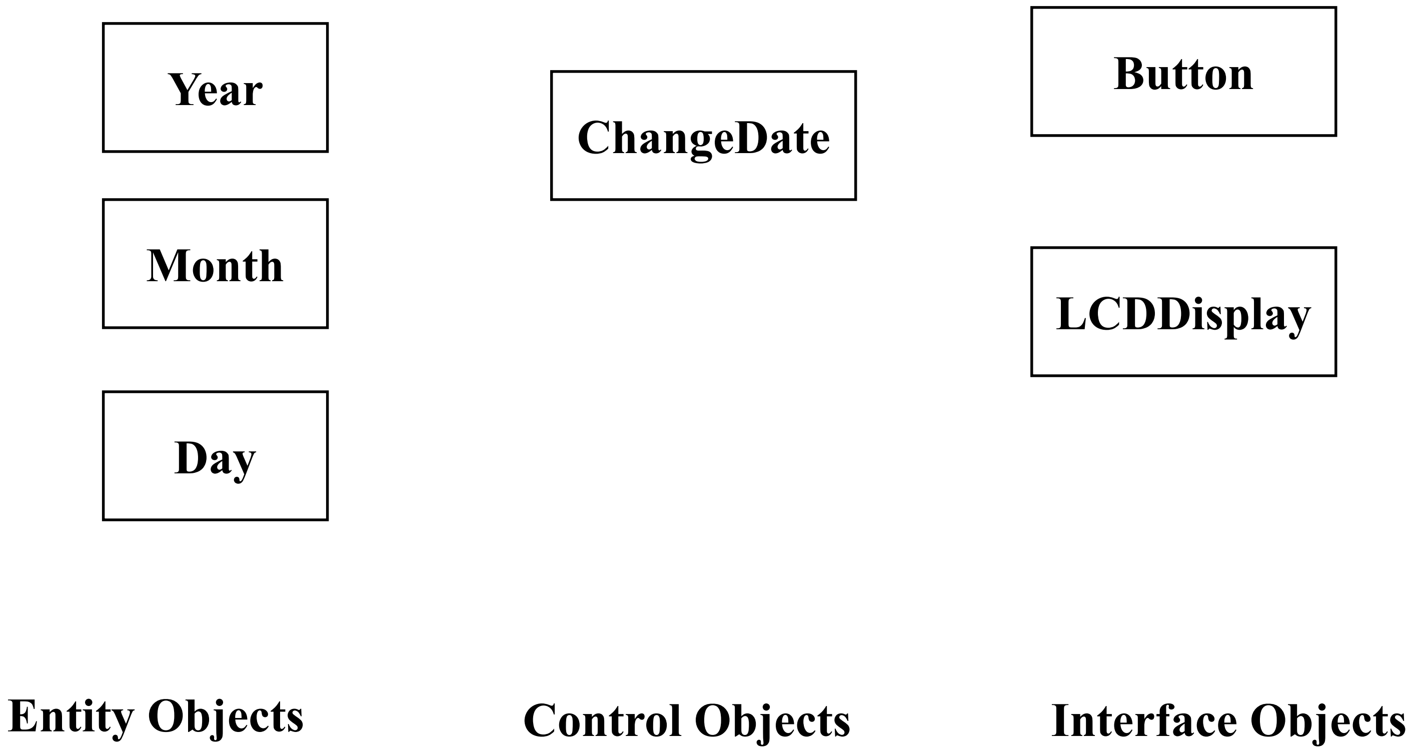
Sequence Diagrams: Entity

- a store or persistence mechanism that captures the information or knowledge in a system
  - class attributes if saved to a data store and can be reloaded, it is termed 'persistent‘
- it is the Model in the Model-View-Controller pattern

Sequence Diagrams: Control

- a stereotyped class that represents a controlling entity or manager
- control organizes and schedules other activities and elements
- it is the controller of the Model-View-Controller pattern

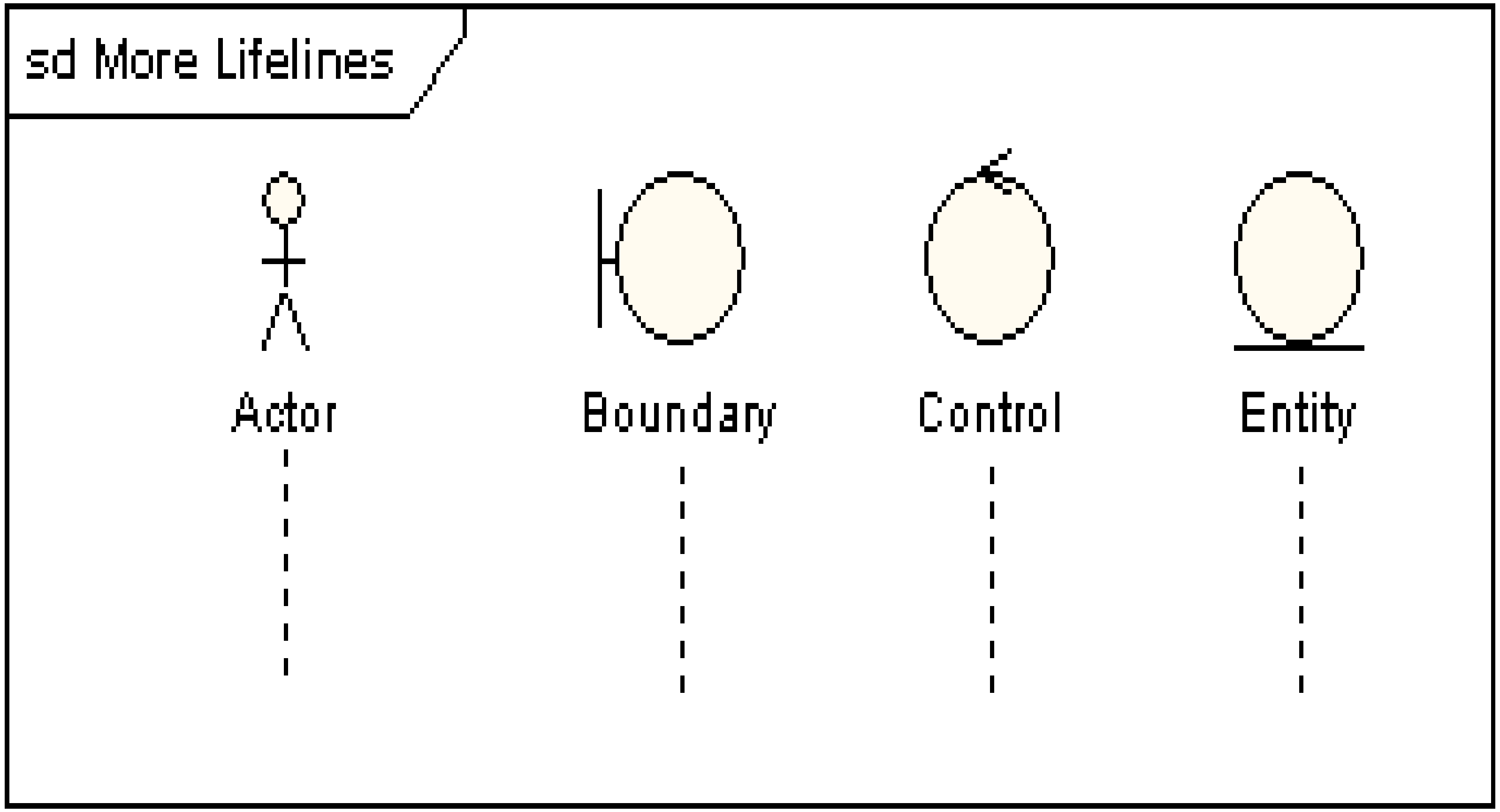
Example: 2BWatch Objects



Sequence Diagram: Lifeline

- a lifeline represents an individual participant in a sequence diagram
- a lifeline will usually have a rectangle containing its object name
- if its name is "self", that indicates that the lifeline represents the classifier which owns the sequence diagram
- sometimes a sequence diagram will have a lifeline with an actor element symbol at its head – usually be the case if the sequence diagram is owned by a use case
- boundary, control and entity elements can also own lifelines

Sequence Diagram: Lifeline …

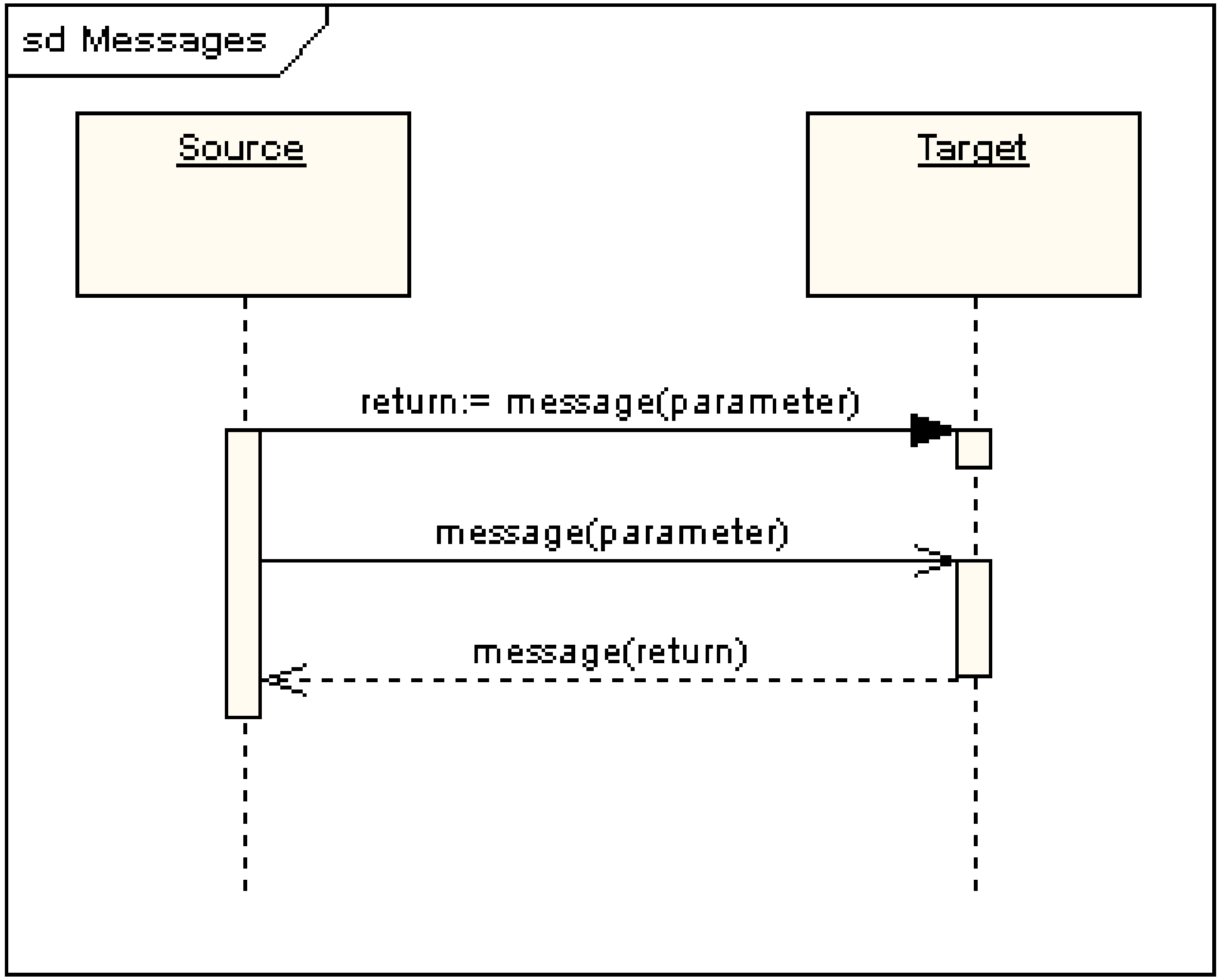


Sequence Diagram: Messages

- displayed as arrows, messages can be complete; lost or found; synchronous or asynchronous; call or signal

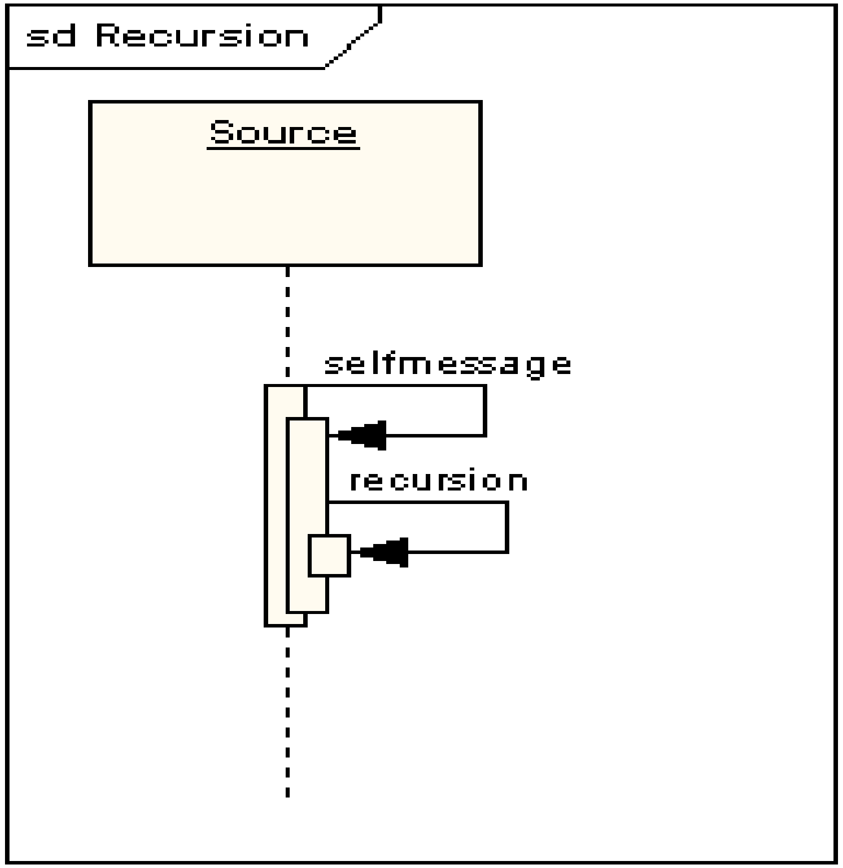
- example diagram
  - the first message is a synchronous message (denoted by the solid arrowhead) complete with an implicit return message
  - the second message is asynchronous (denoted by line arrowhead)
  - the third message is the asynchronous return message (denoted by the dashed line)

Sequence Diagram: Messages …



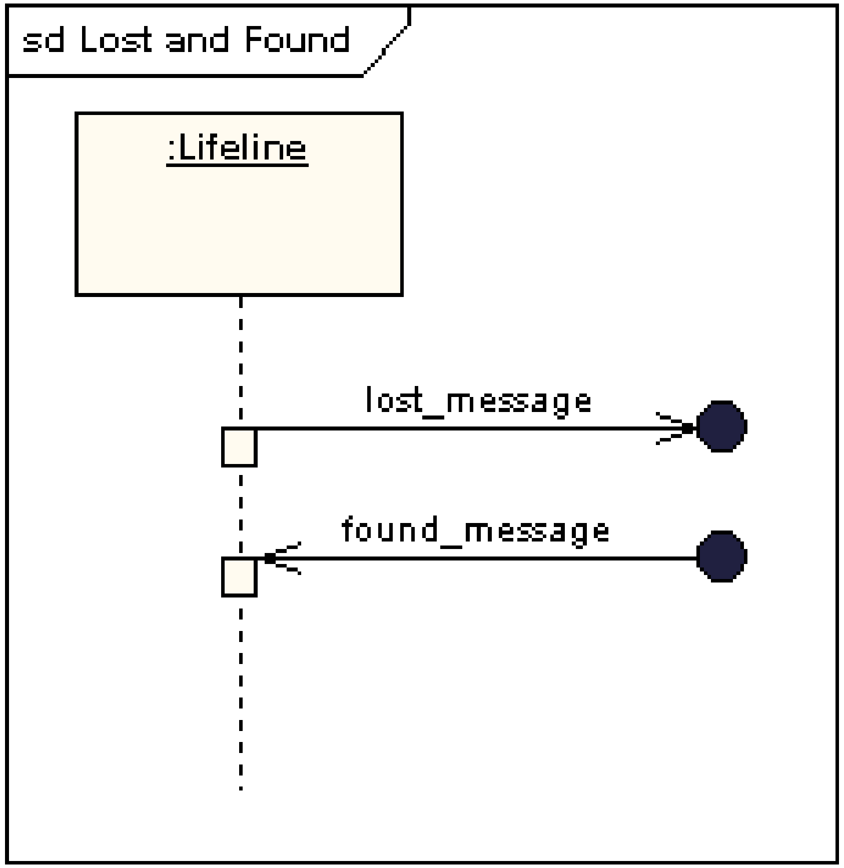
Sequence Diagram: Self Message

- **can represent a recursive call of an operation, or one method calling another method belonging to the same object**
- **It is shown as creating a nested focus of control in the lifeline's execution occurrence**



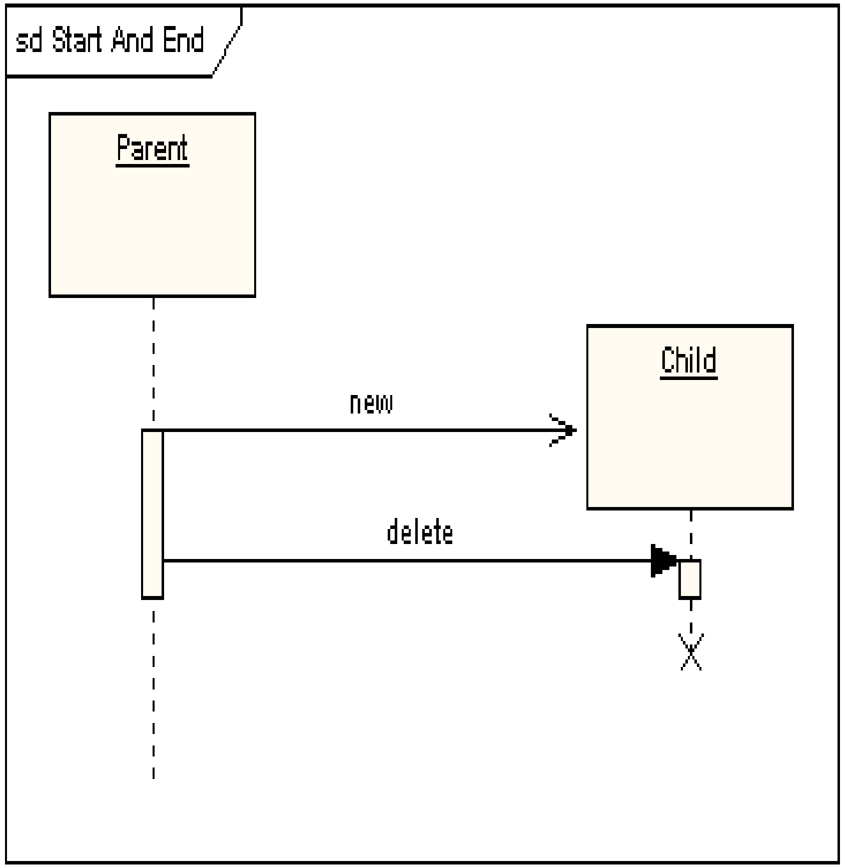
Sequence Diagram: Lost and Found Messages

- lost messages are those that are either sent but do not arrive at the intended recipient, or which go to a recipient not shown on the current diagram
- found messages are those that arrive from an unknown sender, or from a sender not shown on the current diagram
- they are denoted going to or coming from an endpoint element



Sequence Diagram: Lifeline Start and End

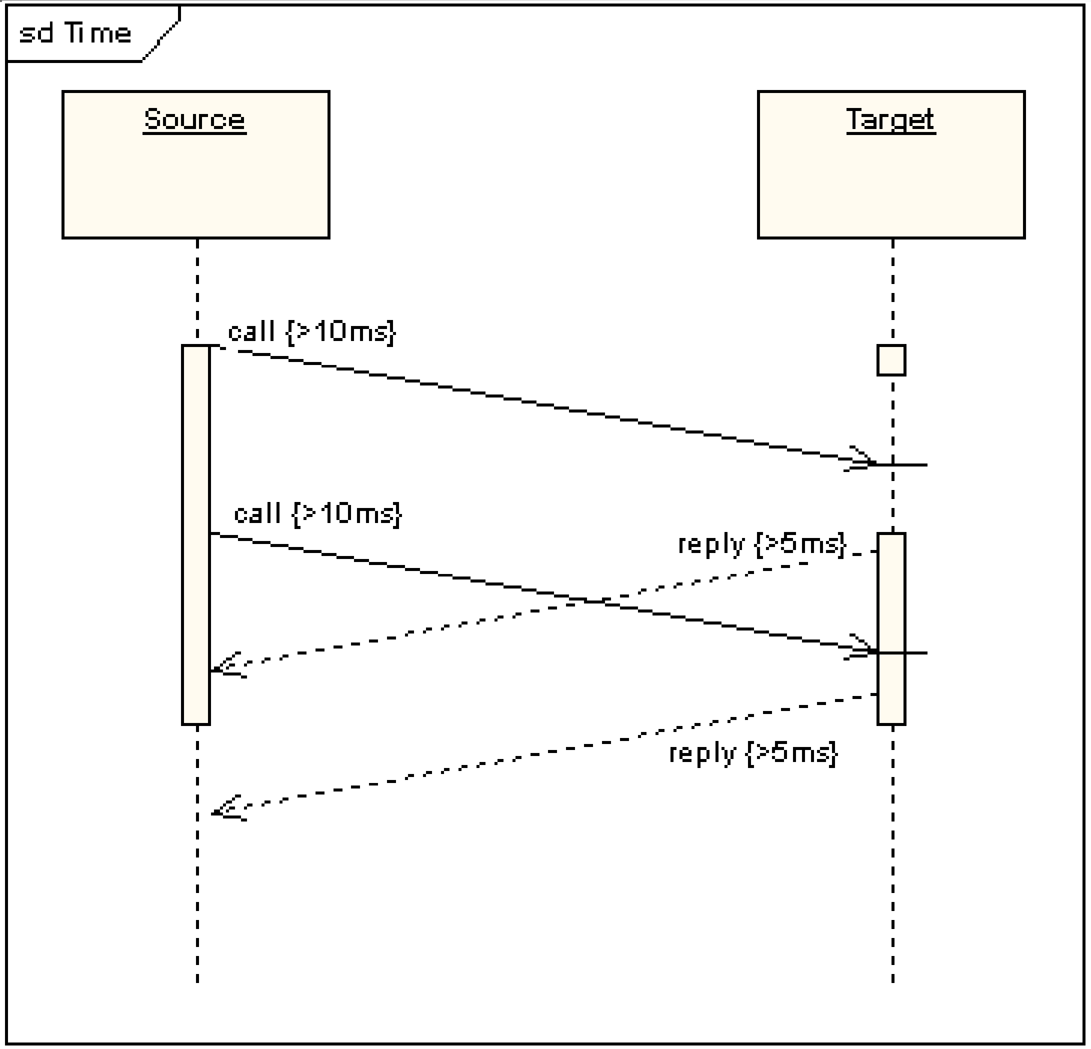
- a lifeline may be created or destroyed during the timescale represented by a sequence diagram
- in the latter case (object destroyed), the lifeline is terminated by a stop symbol, represented as a cross
- in the former case (object created), the symbol at the head of the lifeline is shown at a lower level down the page than the symbol of the object that caused the creation



Sequence Diagram: Duration and Time Constraints

- by default, a message is shown as a horizontal line in a sequence diagram
- since the lifeline represents the passage of time down the screen, when modeling a real-time system, or even a time-bound business process, it can be important to consider the length of time it takes to perform actions
- by setting a duration constraint for a message, the message will be shown as a sloping line

Sequence Diagram: Duration and Time Constraints …



Sequence Diagram: Combined Fragments

- a combined fragment is one or more processing sequence enclosed in a frame and executed under specific named circumstances

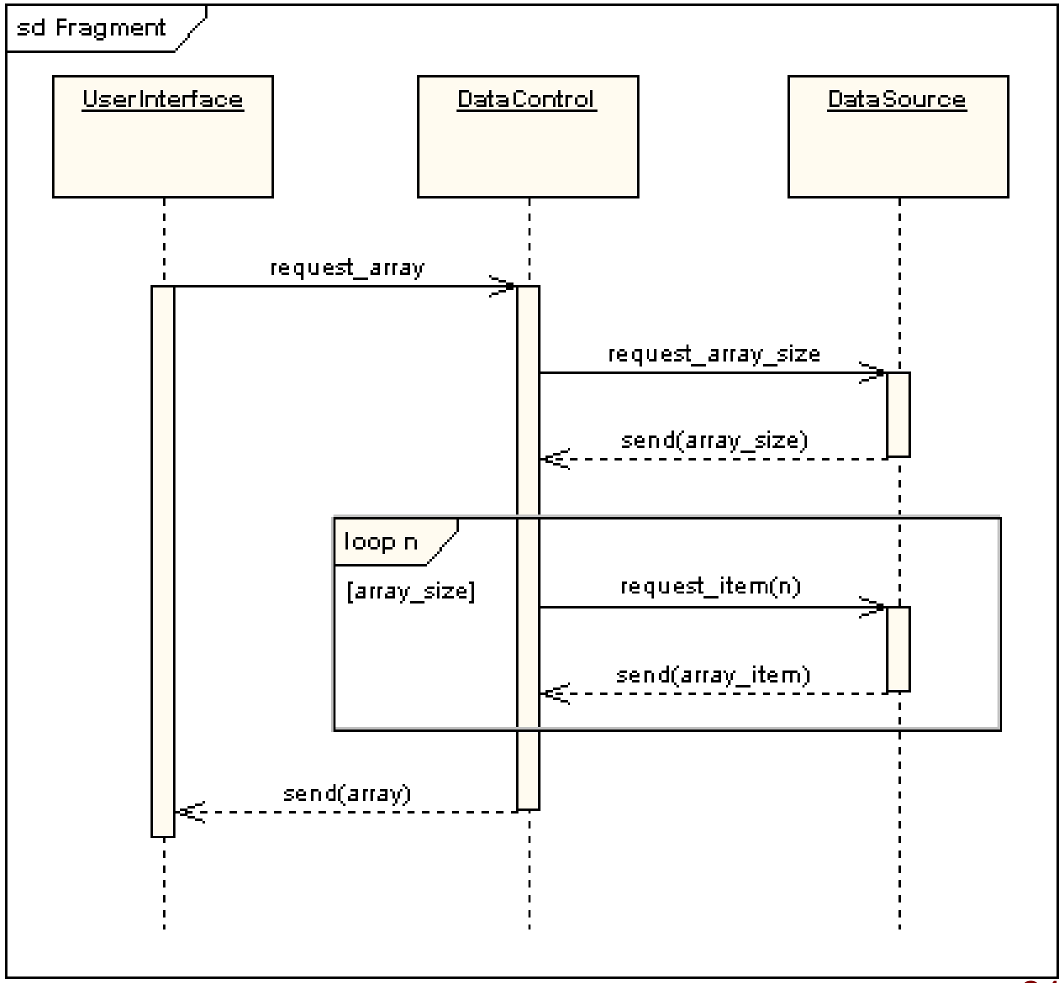
- some fragments available are:
  - alternative fragment (denoted “alt”) models if…then…else constructs
  - option fragment (denoted “opt”) models switch constructs
  - break fragment models an alternative sequence of events that is processed instead of the whole of the rest of the diagram
  - parallel fragment (denoted “par”) models concurrent processing
  - weak sequencing fragment (denoted “seq”) encloses a number of sequences for which all the messages must be processed in a preceding segment before the following segment can start, but which does not impose any sequencing within a segment on messages that don’t share a lifeline

Sequence Diagram: Combined Fragments …

- other fragments available are:
  - strict sequencing fragment (denoted “strict”) encloses a series of messages which must be processed in the given order
  - critical fragment encloses a critical section
  - ignore fragment declares a message or message to be of no interest if it appears in the current context
  - assertion fragment (denoted “assert”) designates that any sequence not shown as an operand of the assertion is invalid
  - loop fragment encloses a series of messages which are repeated

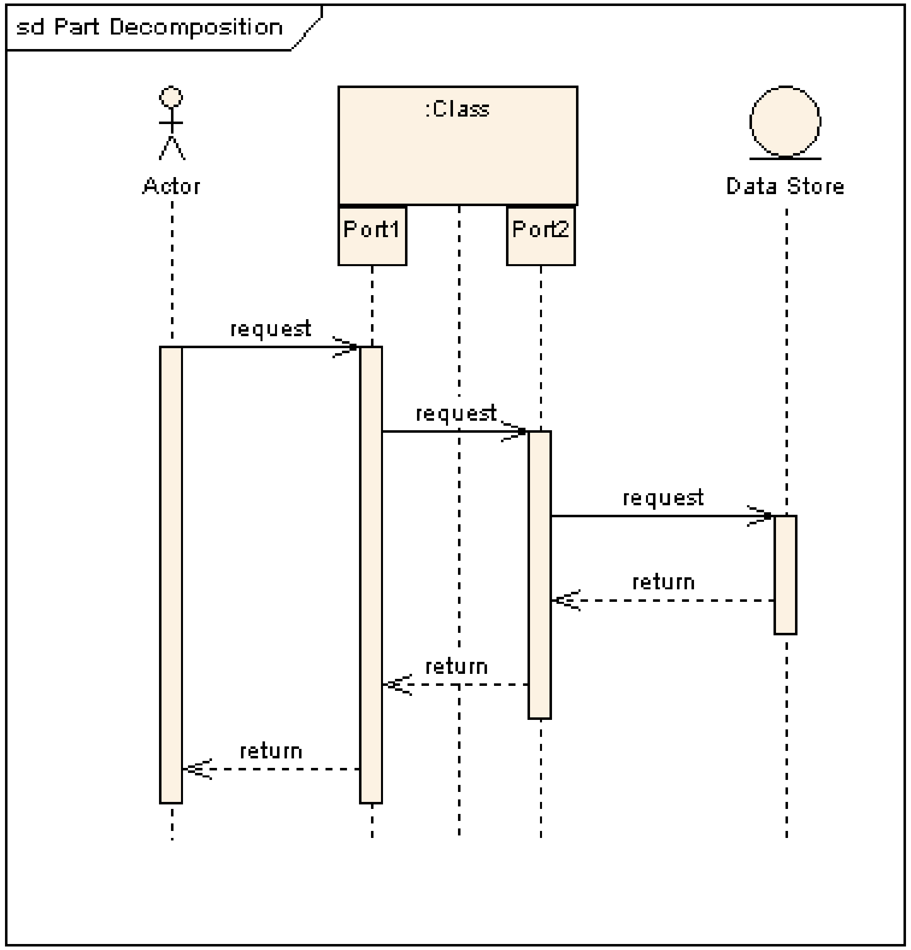
Sequence Diagram: Combined Fragments …

- diagram shows a loop fragment

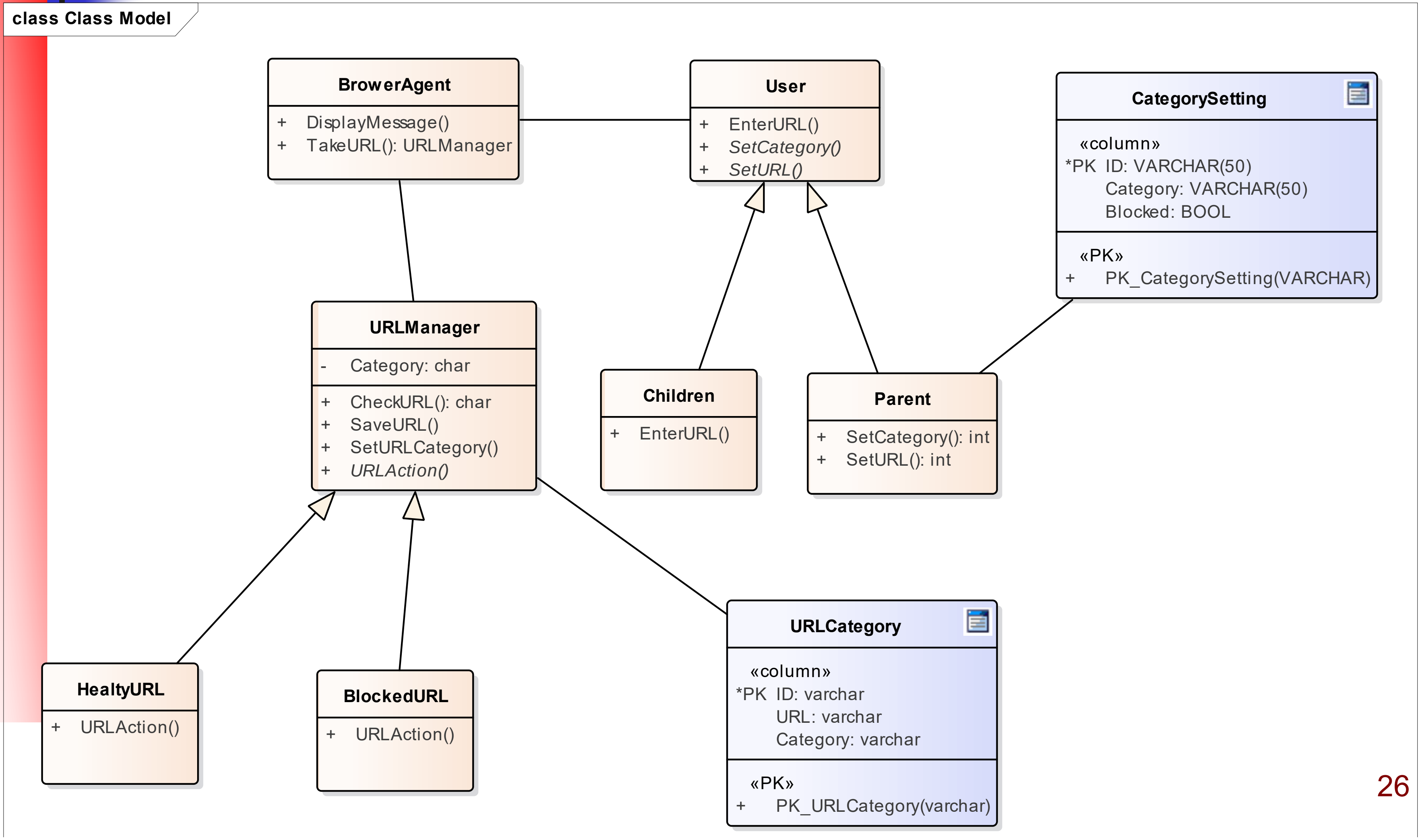


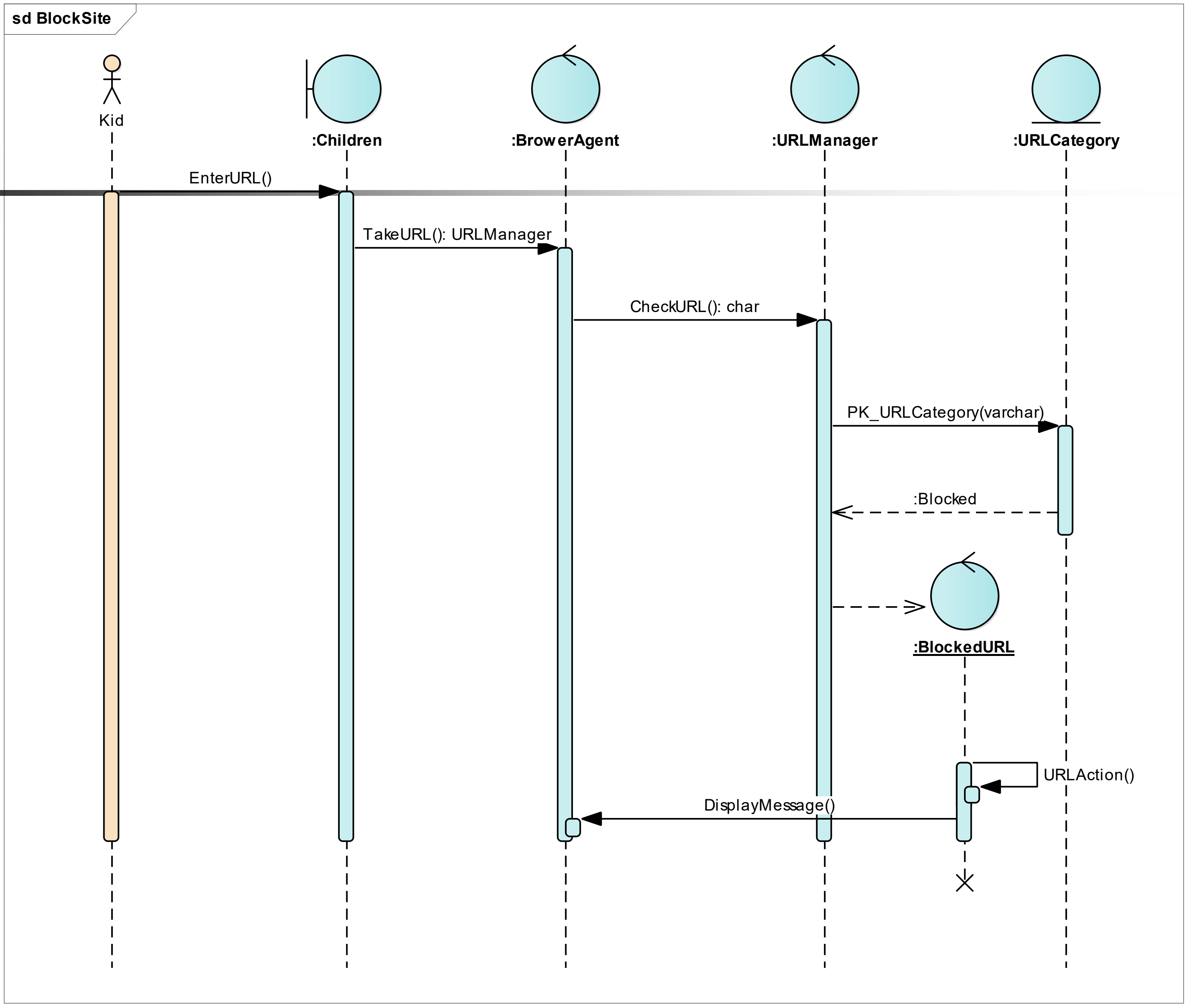
Sequence Diagram: Part Decomposition

- an object can have more than one lifeline coming from it
- this allows for inter- and intra-object messages to be displayed on the same diagram



- Child opens Web browser Chrome
- Child enters www.##########.com in the address box
- E-guard takes the URL and checks the address DB for category
- It shows this URL belongs to a blocked category set by parent
- E-guard blocks the site and shows a warning message in browser



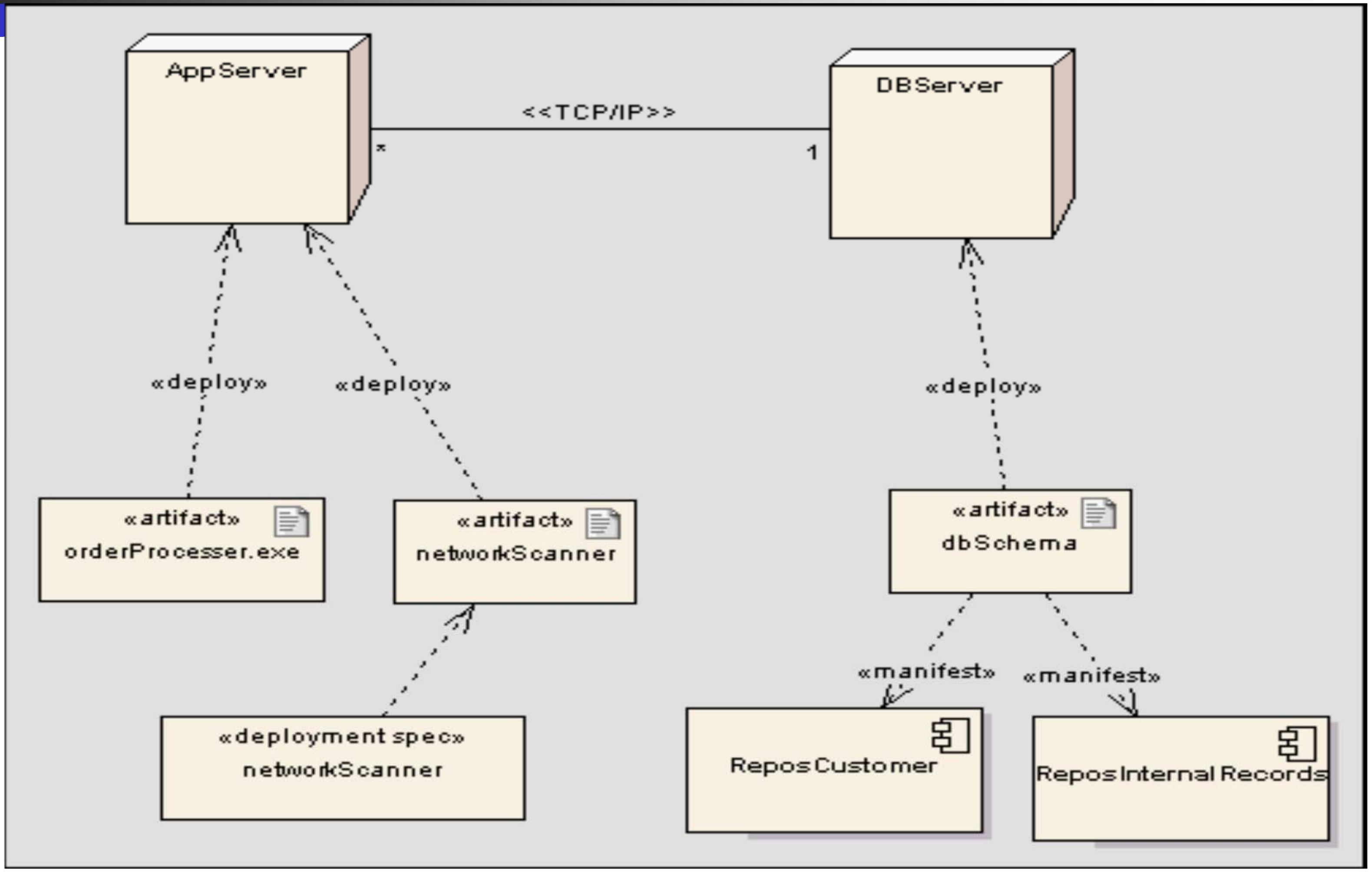

sd BlockSite
Kid
:Children
:BrowerAgent
:URLManager
:URLCategory
EnterURL()
TakeURL(): URLManager
CheckURL(): char
PK_URLCategory(varchar)
:Blocked
:BlockedURL
URLAction()
DisplayMessage()

Deployment Diagrams

Deployment Diagrams

- models the run-time architecture of a system
- shows the configuration of the hardware elements (nodes) and shows how software elements and artifacts are mapped onto those nodes
    - shows how and where the system will be deployed
- physical machines and processors are reflected as nodes, and the internal construction can be depicted by embedding nodes or artifacts
- as artifacts are allocated to nodes to model the system's deployment, the allocation is guided by the use of deployment specifications

Deployment Diagrams: Example 1

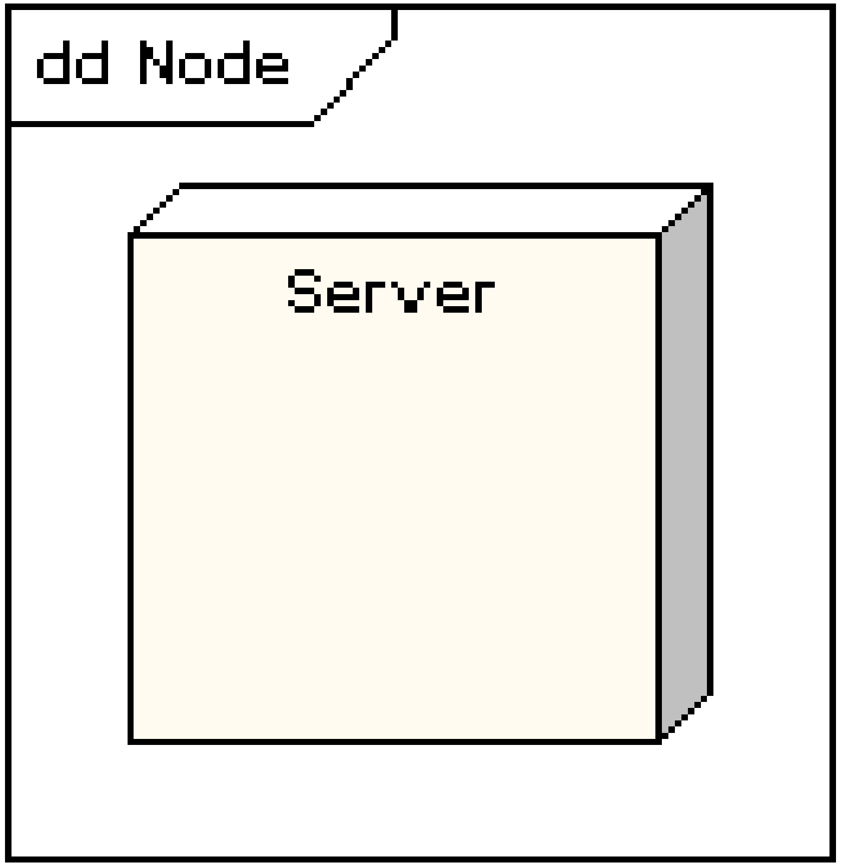


Deployment Diagrams: Elements

- node
- node instance
- node stereotypes
- artifact
- interface
- association
- association class
- node as a container
- package
- component
- generalize
- realize
- deployment
- deployment spec
- dependency
- trace
- object flow
- nesting

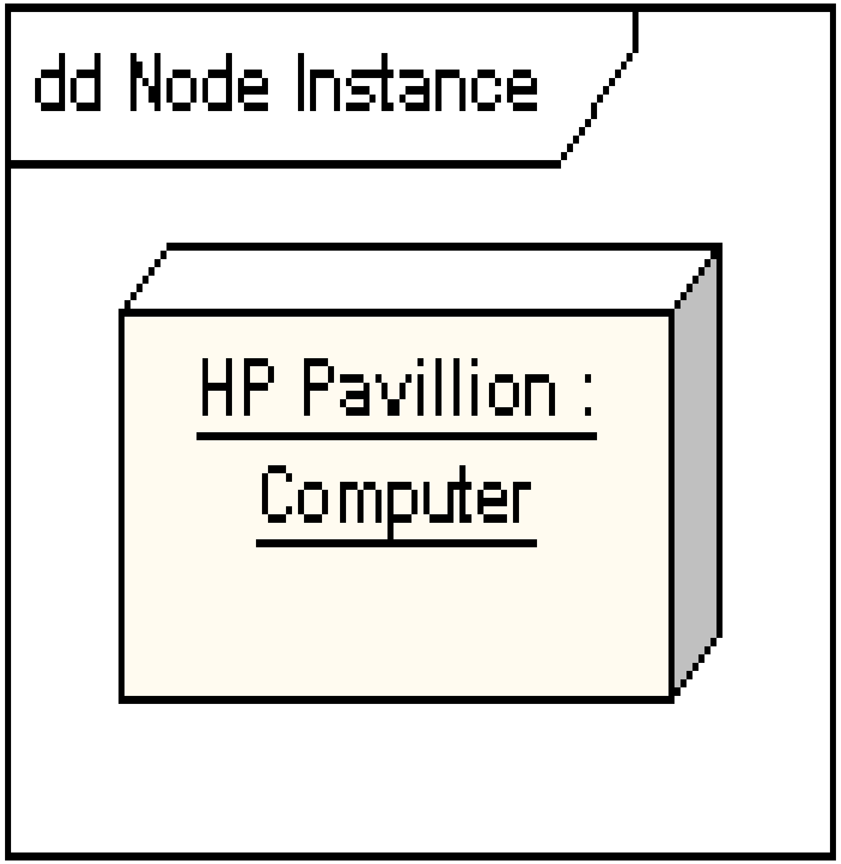
Deployment Diagrams: Node

- a node is either a hardware or software element
- it is shown as a three-dimensional box shape, as shown below



Deployment Diagrams: Node Instance

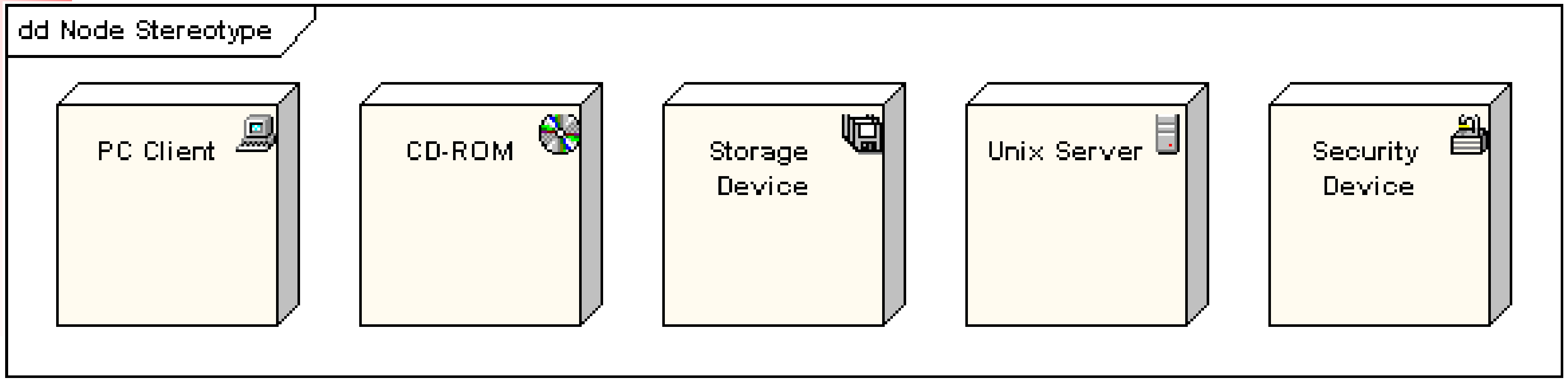
- a node instance can be shown on a diagram
- an instance can be distinguished from a node by the fact that its name is underlined and has a colon before its base node type
- an instance may or may not have a name before the colon
- the following diagram shows a named instance of a computer



Deployment Diagrams: Node Stereotypes

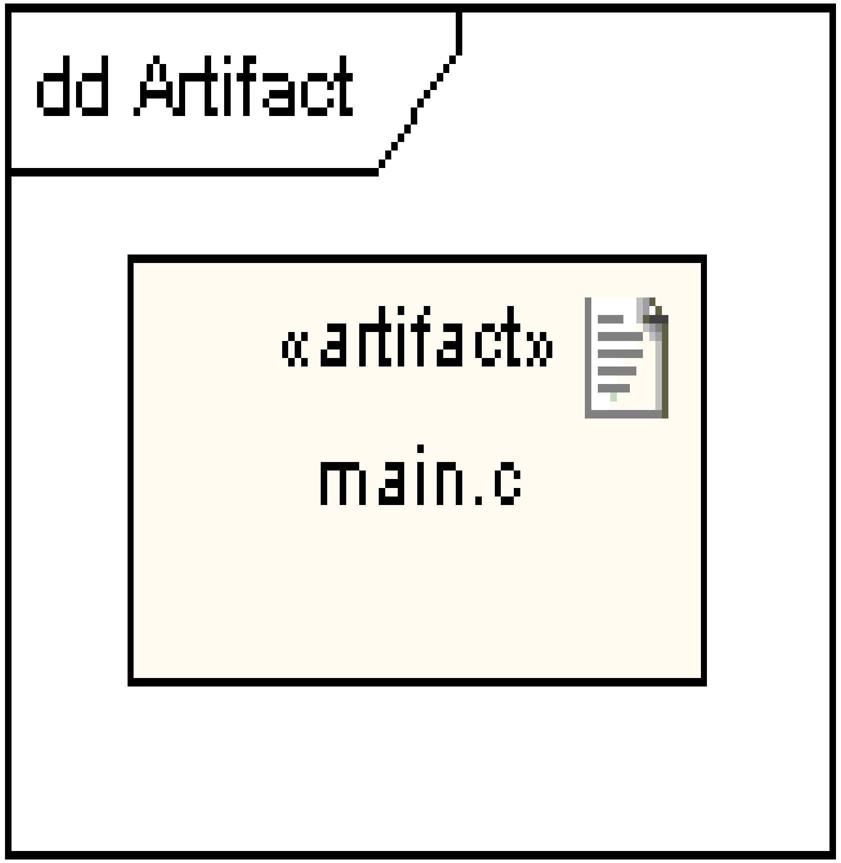
- a number of standard stereotypes are provided for nodes, namely
  - «cdrom», «cd-rom», «computer», «disk array», «pc», «pc client», «pc server», «secure», «server», «storage», «unix server», «user pc»

- these will display an appropriate icon in the top right corner of the node symbol



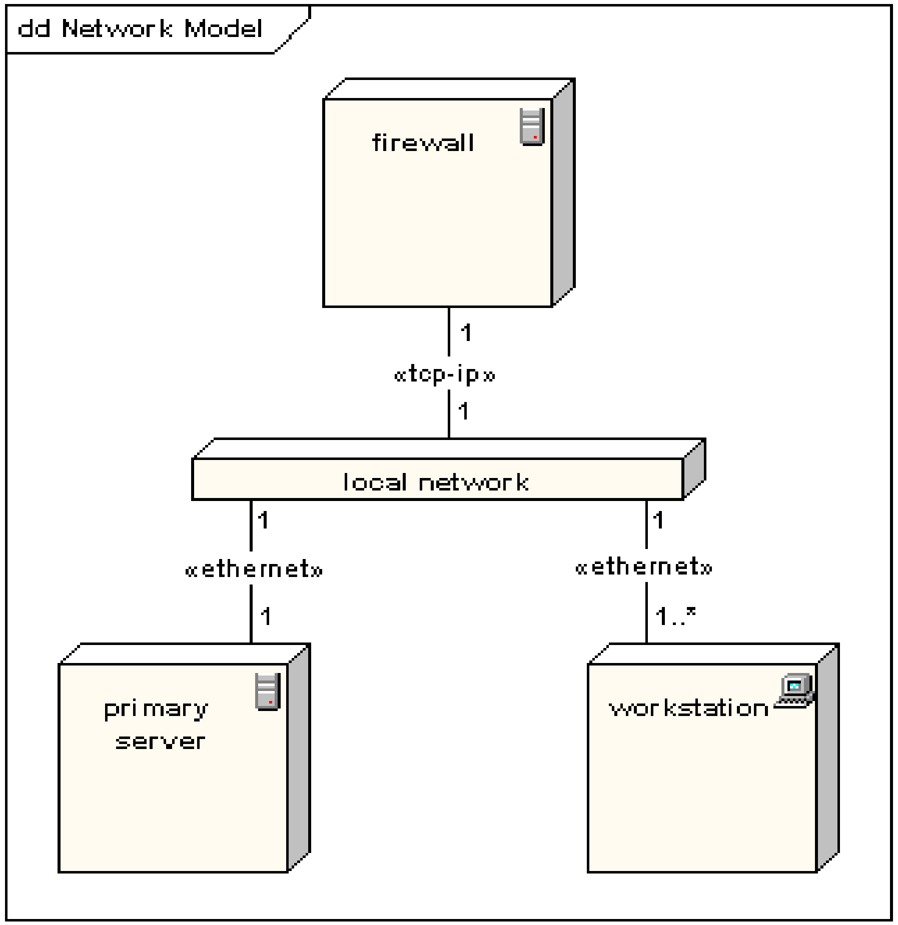
Deployment Diagrams: Artifact

- an artifact is a product of the software development process
- may include process models (e.g. use case models, design models etc), source files, executables, design documents, test reports, prototypes, user manuals, etc.
- denoted by a rectangle showing the artifact name, the «artifact» keyword and a document icon



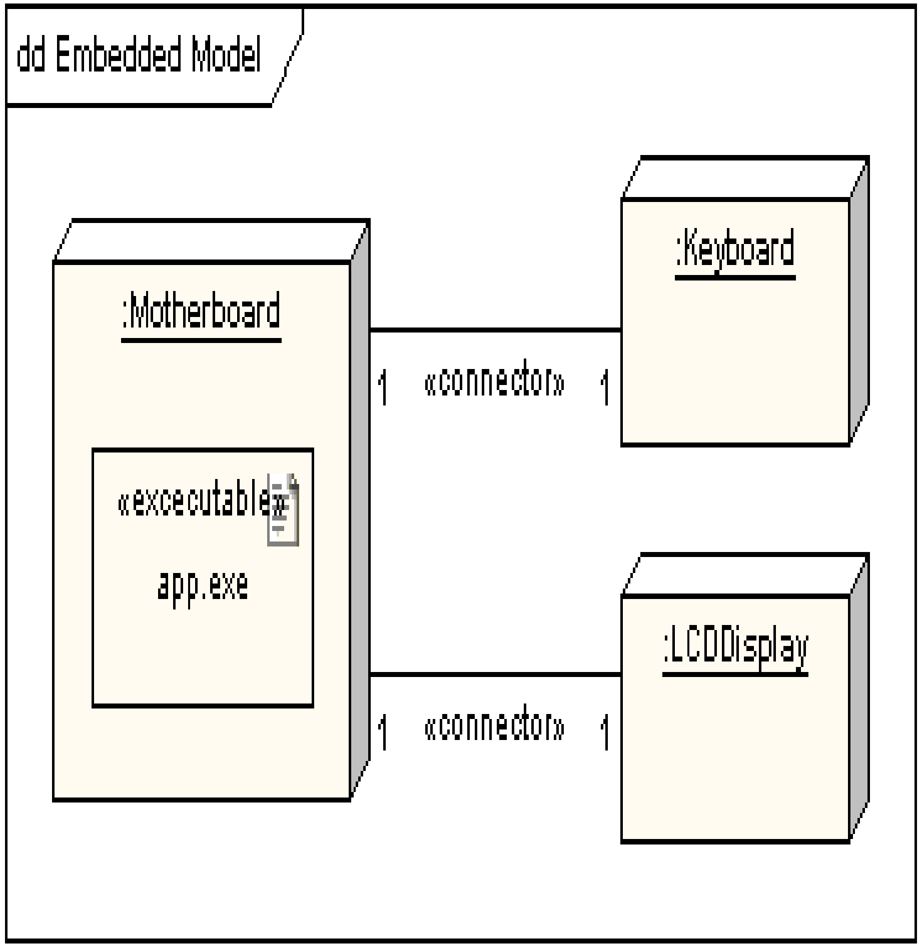
Deployment Diagrams: Association

- an association in a deployment diagram represents a communication path between nodes
- following diagram shows a deployment diagram for a network, depicting network protocols as stereotypes, and multiplicities at the association ends



Deployment Diagrams: Node as a Container

- a node can contain other elements, such as components or artifacts
- following diagram shows a deployment diagram for part of an embedded system, depicting an executable artifact as being contained by the motherboard node

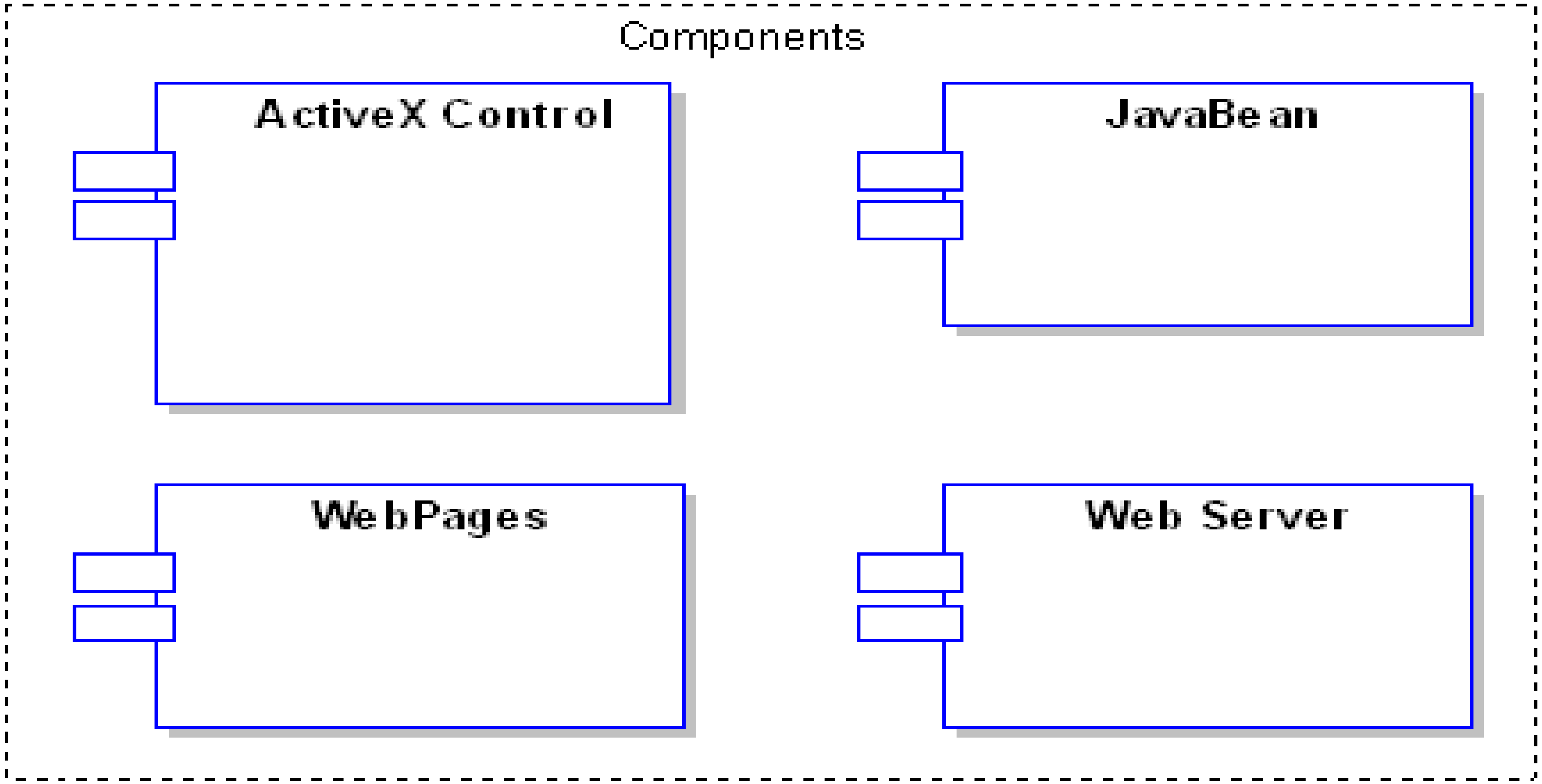


Component Models

Component Model

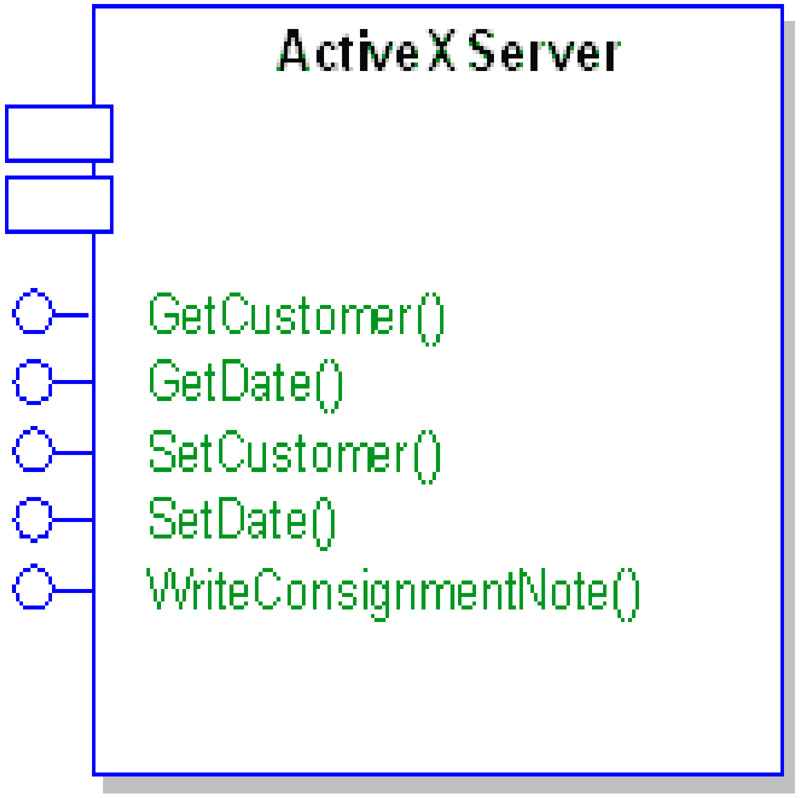
- illustrates the software components that will be used to build the system
- components may be built up from the class model and written from scratch for the new system
- components may be brought in from other projects and 3rd party vendors
- components are high level aggregations of smaller software pieces, and provide a 'black box' building block approach to software construction

Component Model: Example

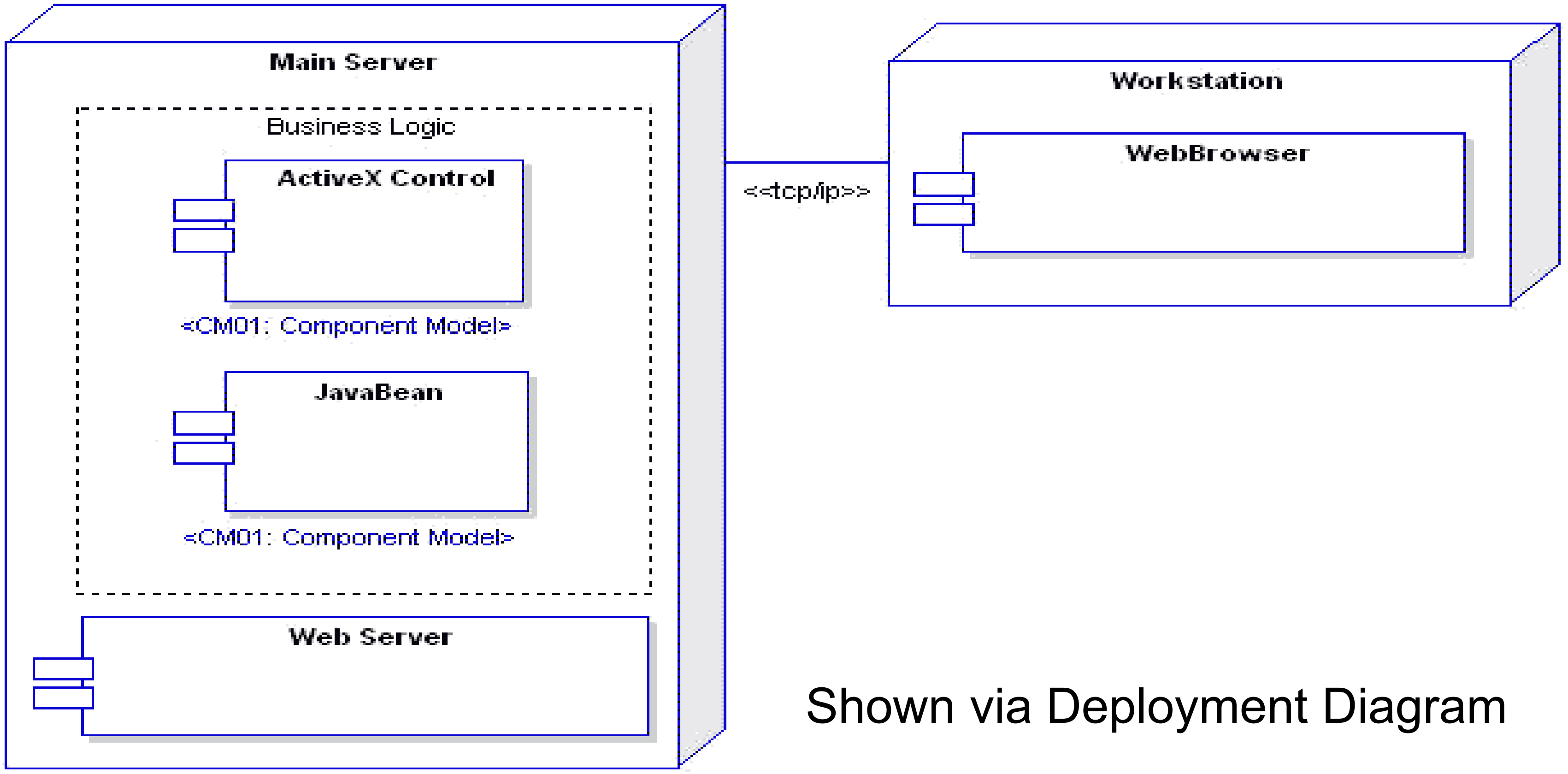


Component Model: Interfaces

- components may also expose interfaces
- these are the visible entry points or services that a component is advertising and making available to other software components and classes
- typically a component is made up of many internal classes and packages of classes
  - it may even be assembled from a collection of smaller components



Component Model: Components and Nodes



Shown via Deployment Diagram

Component Model: Other Items

- requirements
  - components may have requirements attached to indicate their contractual obligations, i.e. what service they will provide in the model
  - requirements help document the functional behavior of software elements

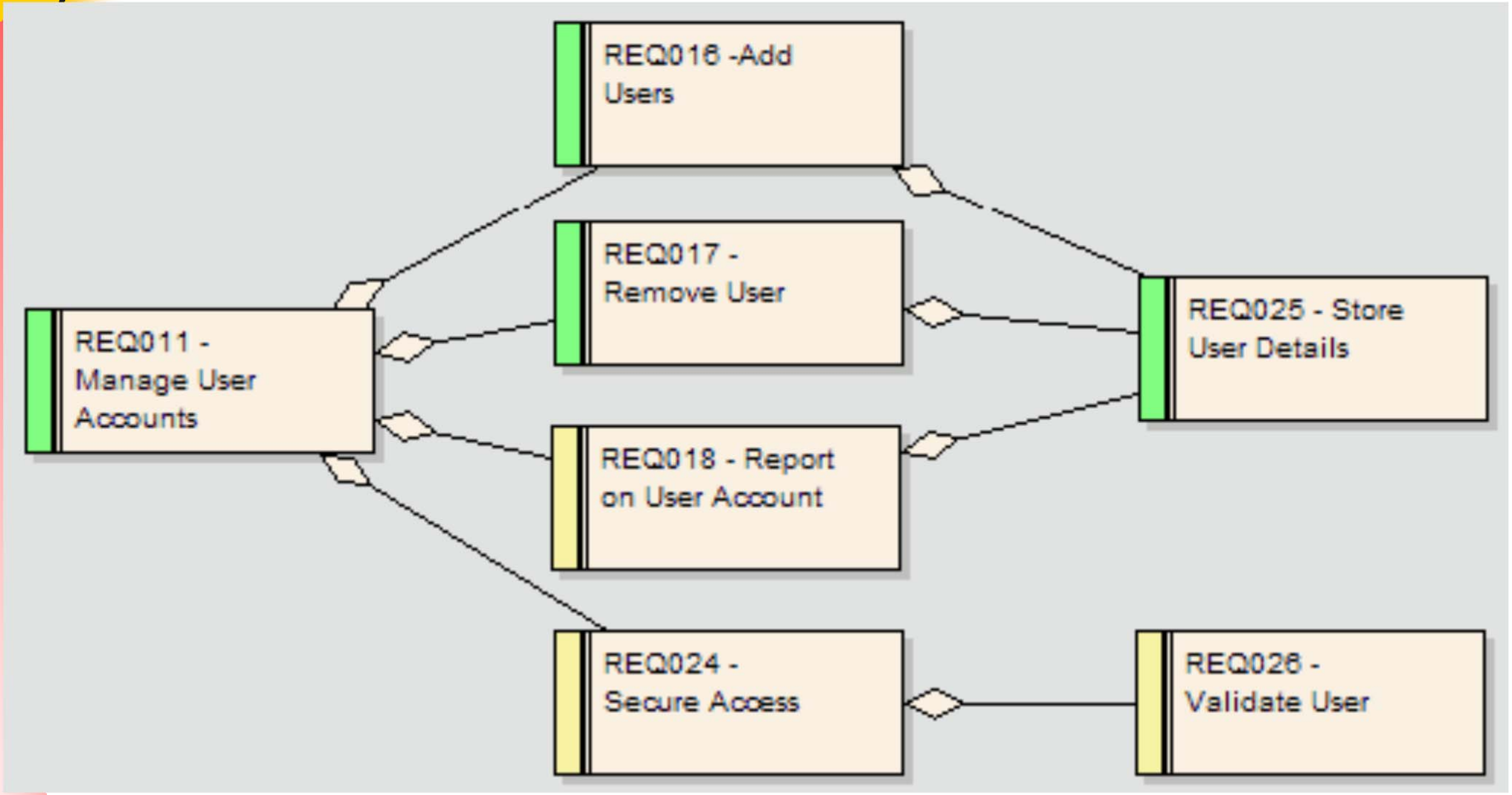
- constraints
  - components may have constraints attached which indicate the environment in which they operate
  - pre-conditions specify what must be true before a component can perform some function
  - post-conditions indicate what will be true after a component has done some work
  - invariants specify what must remain true for the duration of the components lifetime

Component Model: Other Items …

- scenarios
  - scenarios are textual/procedural descriptions of an object's actions over time and describe the way in which a component works
  - multiple scenarios may be created to describe the basic path (a perfect run through) as well as exceptions, errors and other conditions

Requirements Model

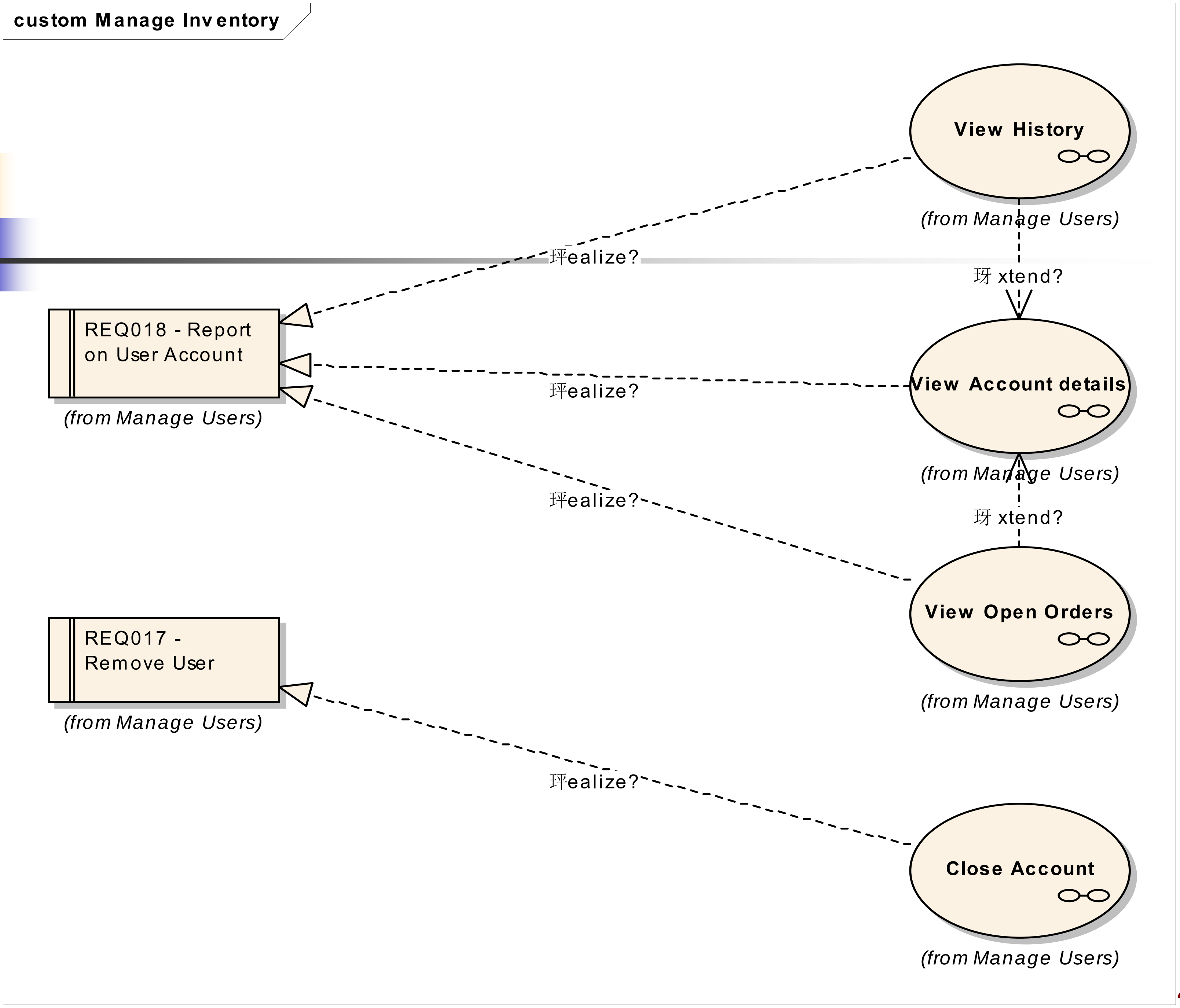
- a custom diagram used to describe a system's requirements or features as a visual model
- requirement elements may then be linked back to use cases and components in the system to illustrate how a particular system requirement is met
- requirements models enable for traceability between specifications and design requirements, and the model elements which realize them


REQ016 -Add Users
REQ017 - Remove User
REQ011 - Manage User Accounts
REQ018 - Report on User Account
REQ025 - Store User Details
REQ024 - Secure Access
REQ026 - Validate User

Traceability Modeling

Traceability

- a traceability diagram shows the mapping from requirements to specific use cases
- traceability diagram is simply a set of elements from the requirements section and the use case section dragged onto this diagram
- traceability relationships can be created using the Relationship Matrix
  - the relationship matrix is accessible from the main menu -under: View | Relationship Matrix
  - you can create the relationship matrix from a traceability diagram


custom Manage Inventory
View History
(from Manage Users)
玶ealize?
玕 xtend?
REQ018 - Report on User Account
(from Manage Users)
玶ealize?
View Account details
(from Manage Users)
玶ealize?
玕 xtend?
View Open Orders
(from Manage Users)
REQ017 - Remove User
(from Manage Users)
玶ealize?
Close Account
(from Manage Users)

User Interface Diagrams

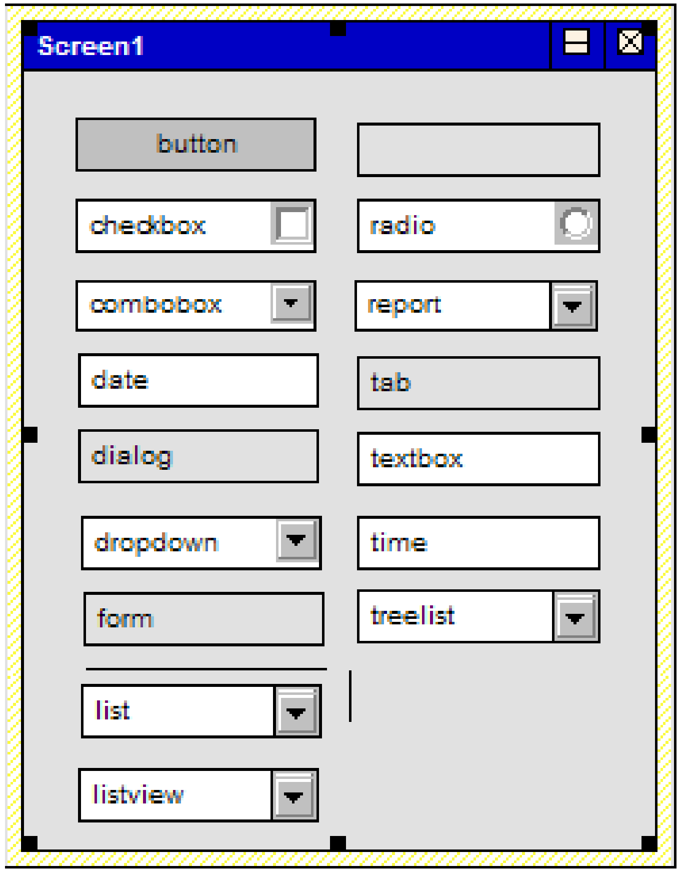
- used to visually mock-up a system's user interface using forms, controls and labels.
- a user interface diagram is a custom diagram used to describe the visual mock-up of a system's user interface

User Interface Diagrams: User Interface Items

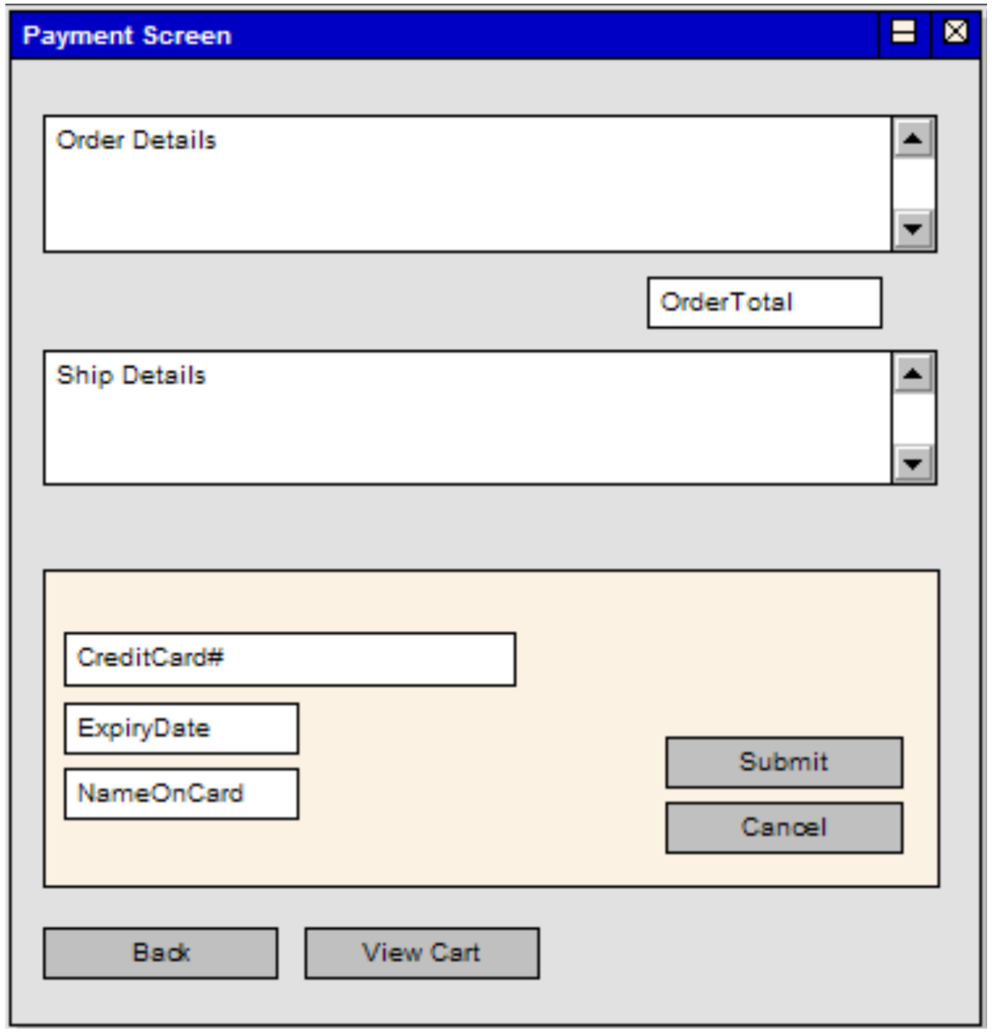
- User Interface Group enables you to create graphical user interface diagrams
- a **Screen** element represents a graphical user interface. You will be able to place GUI elements onto the screen element
- **GUI Elements** are placed onto the screen element to build up a graphical user interface diagram. There are different stereotypes that represent different elements such as buttons and combo boxes
- an **Object** is an instance of a class

User Interface Diagrams: GUI Elements

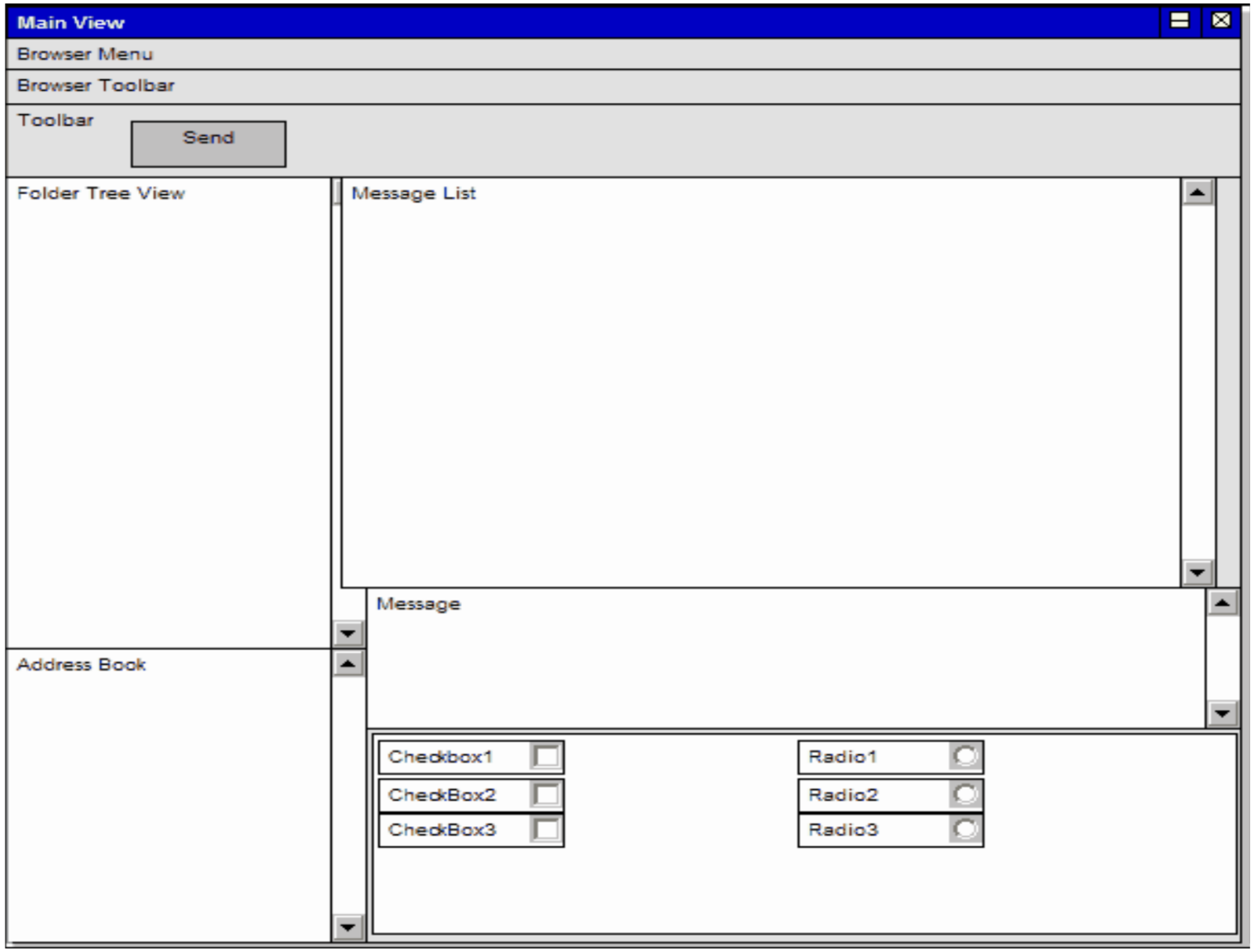
- button
- checkbox
- combo box
- date
- dialog
- dropdown
- form
- list
- listview
- radio
- report
- hline
- panel
- tab
- textbox
- time
- treelist
- vline



User Interface Diagrams: Example 1



User Interface Diagrams: Example 2


Main View
Browser Menu
Browser Toolbar
Toolbar
Send
Folder Tree View
Message List
Message
Address Book
Checkbox1
CheckBox2
CheckBox3
Radio1
Radio2
Radio3