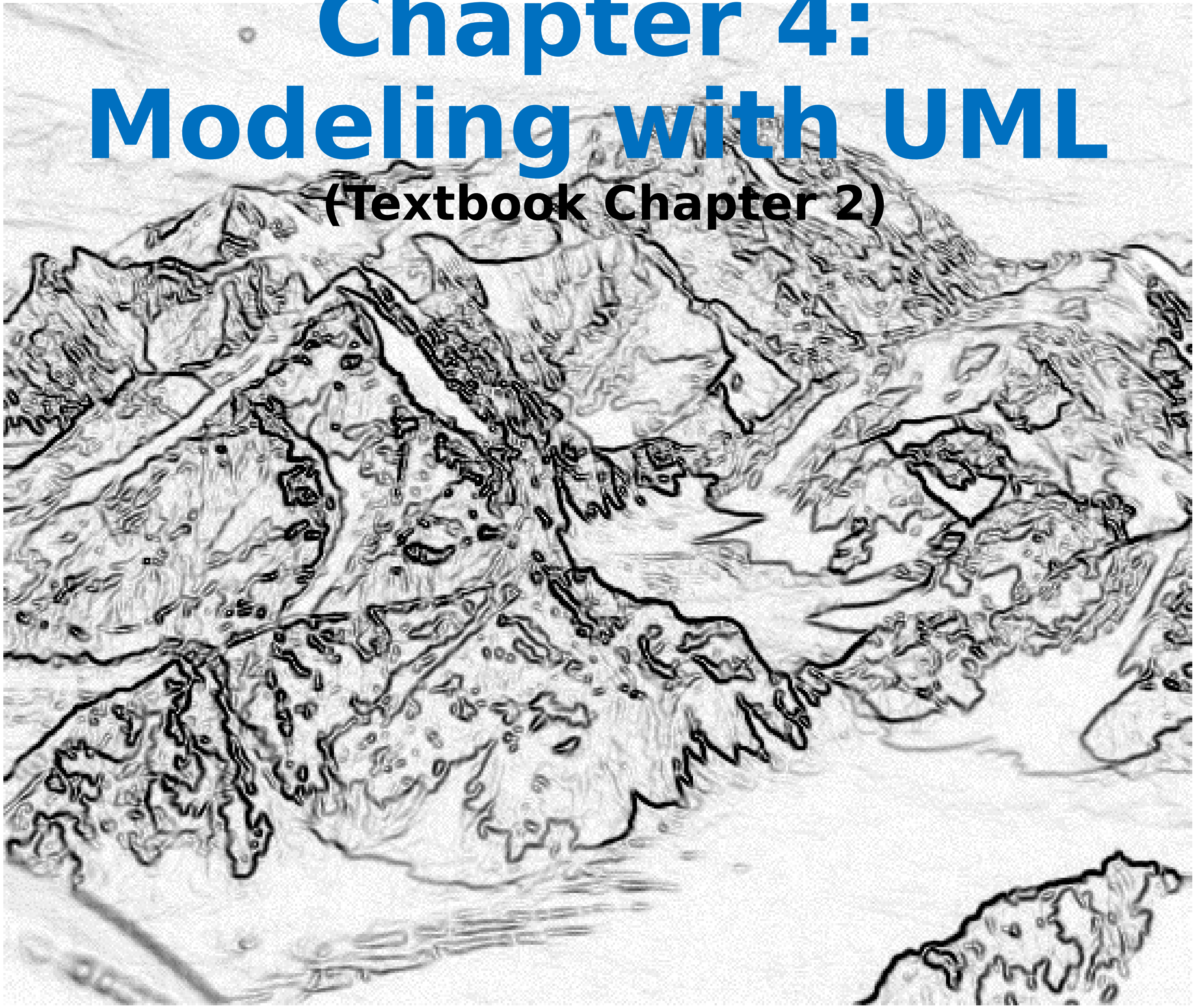
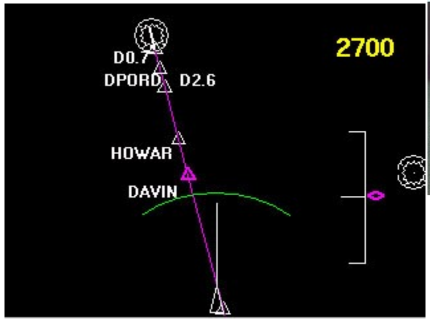


Chapter 4: Modeling with UML

(Textbook Chapter 2)

Object-oriented Modeling

Application Domain (Phenomena)

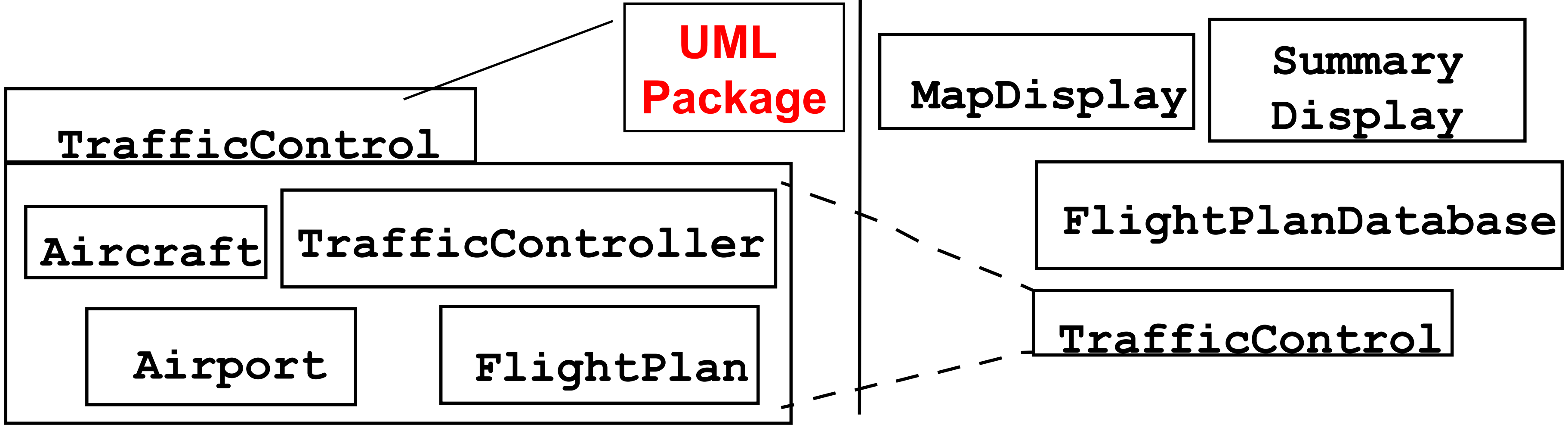


```
us;PETM          PetsMart Inc #                               Common Stock
-  15.8500 Dn    0.0600    (Q)14.34    H  16.1000  L  15.6600  V  1,011,200

E/MM   TIME        -BID-     SIZE(H)     E/MM   TIME      -ASK-     SIZE(H)
(Q)    14.34   +   15.8500     10        (Q)    14.34     15.8700     1
INCA   14.34       15.8500     10        INCA   14.34     15.8700     1
BRUT   14.30       15.8500      7        JRCO   14.28     15.9000     9
ISLD   14.30       15.8500      2        ARCA   14.24     15.9000     5
MLCO   14.34       15.8500      1        SBSH   14.12     15.9100     1
LCAM   14.18       15.8300     10        REDI   14.34     15.9100     1
NITE   14.23       15.8200      1        SLKC   14.18     15.9400     1
JRCO   14.28       15.8300      1        MLCO   14.34     15.9500    10
UBSW   13.57       15.7900      8        FLTT   12.22     15.9500     1
LTCO   14.33       15.7500      5        ISLD   14.30     15.9600    10
SCMB   14.19       15.7400     10        MONT   12.04     15.9600     1
CWCO   12.58       15.7000      1        SCMB   14.19     15.9800     1
REDI   14.34       15.6800      5        UBSW   13.57     16         20
CANT    9.00       15.6600      1        GSCO   12.07     16         10
WCHV   10.10       15.6600      1        BTRD   14.30     16          6

   28-May-02 14:34 NYC                            (c)REUTERS
```

Solution Domain (Phenomena)

System Model (Concepts) *(Analysis)*

System Model (Concepts) *(Design)*

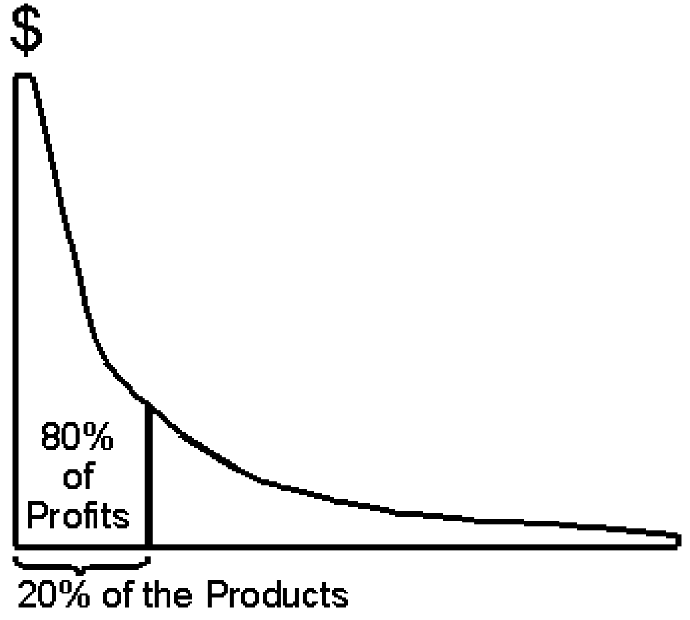


What is UML?

- UML (Unified Modeling Language)
  - Nonproprietary standard for modeling software systems, OMG
  - Convergence of notations used in object-oriented methods
    - OMT (James Rumbaugh and collegues)
    - Booch (Grady Booch)
    - OOSE (Ivar Jacobson)
- Current Version: UML 2.2
  - Information at the OMG portal http://www.uml.org/
- Commercial tools: Rational (IBM),Together (Borland), Visual Architect (business processes, BCD)
- Open Source tools: ArgoUML, StarUML, Umbrello

UML: First Pass

- You can solve 80% of the modeling problems by using 20 % UML
- We teach you those 20%



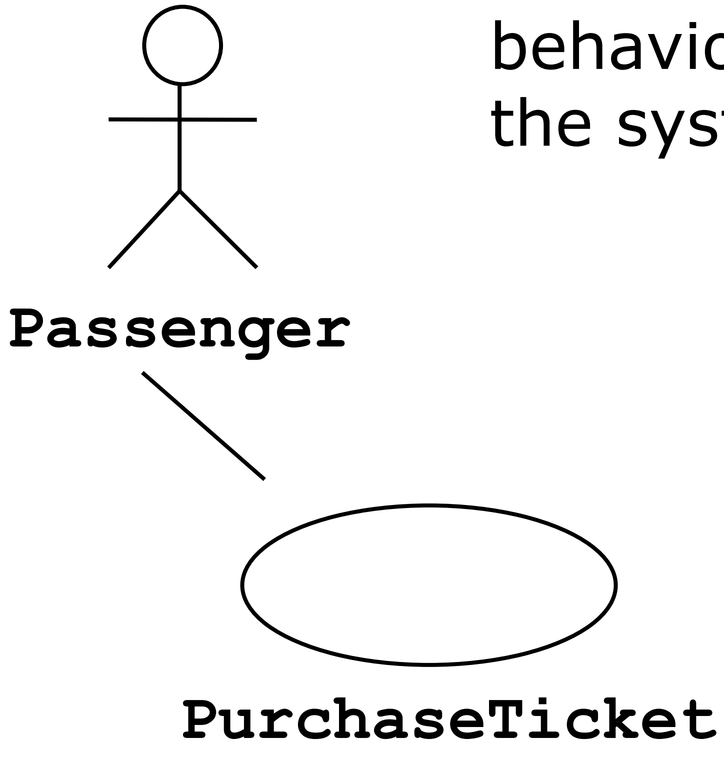
UML First Pass

- Use case diagrams
  - Describe the functional behavior of the system as seen by the user
- Class diagrams
  - Describe the static structure of the system: Classes, attributes, methods, relationships between classes
- Sequence diagrams
  - Describe the dynamic behavior between objects of the system
- Statechart diagrams
  - Describe the dynamic behavior of an individual (/group of) object
- Activity diagrams
  - Describe the dynamic behavior of a system, in particular the workflow.

UML Core Conventions

- All UML Diagrams denote graphs of nodes and edges
  - Nodes are entities and drawn as rectangles or ovals
  - $\boxed{\text{Rectangles}}$ denote classes or instances
  - Ovals denote functions
- Names of Classes are not underlined
  - `SimpleWatch`
  - `FireFighter`
- Names of Instances are underlined
  - $\underline{\texttt{myWatch:SimpleWatch}}$
  - $\underline{\texttt{joe:FireFighter}}$
- An edge between two nodes denotes a relationship between the corresponding entities

UML Use Case Diagrams



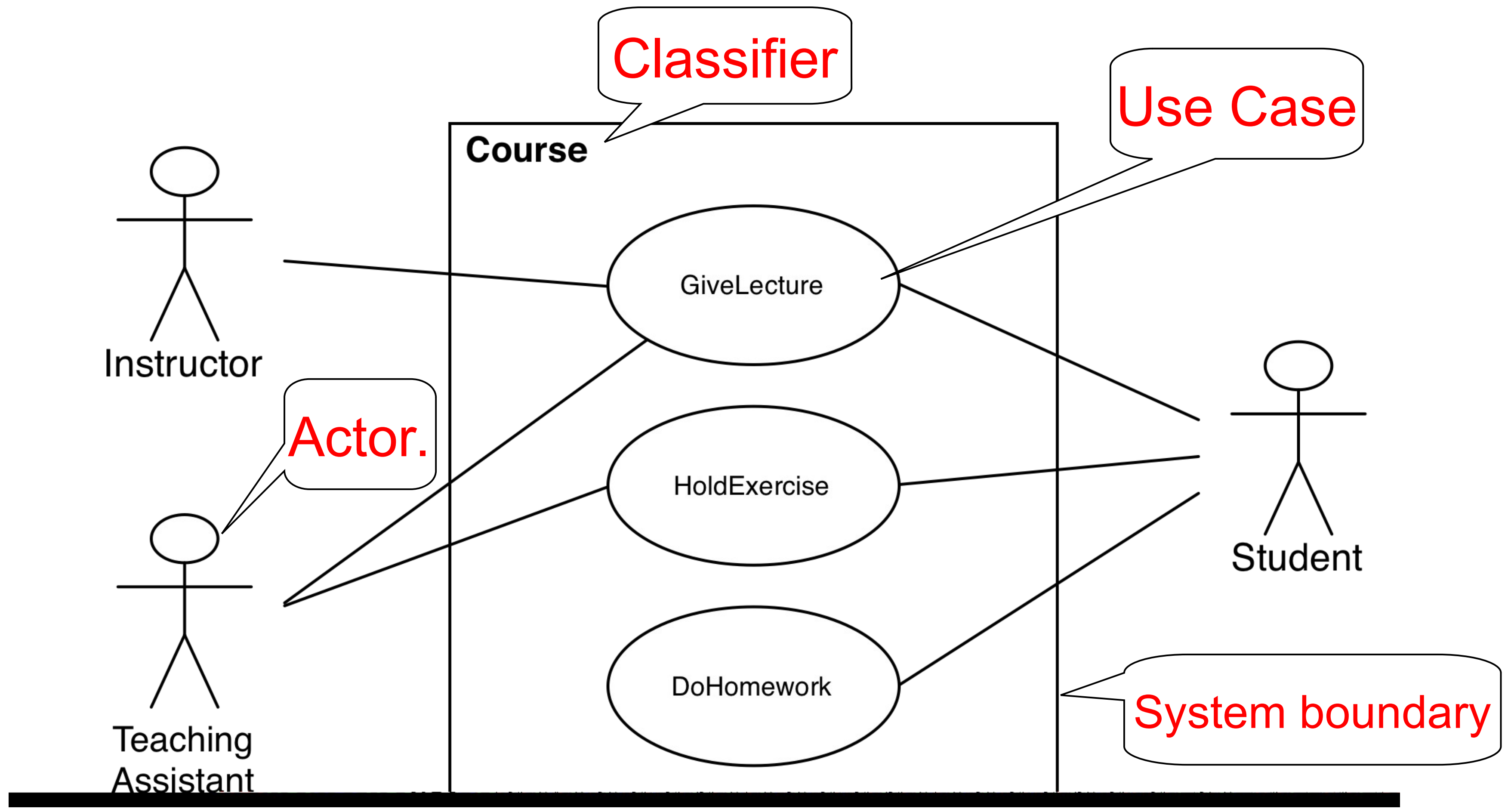
Used during requirements elicitation and analysis to represent external behavior ("visible from the outside of the system")

An ***Actor*** represents a role, that is, a type of user of the system

A ***use case*** represents a class of functionality provided by the system

Use case model:
The set of all use cases that completely describe the functionality of the system + Actors.

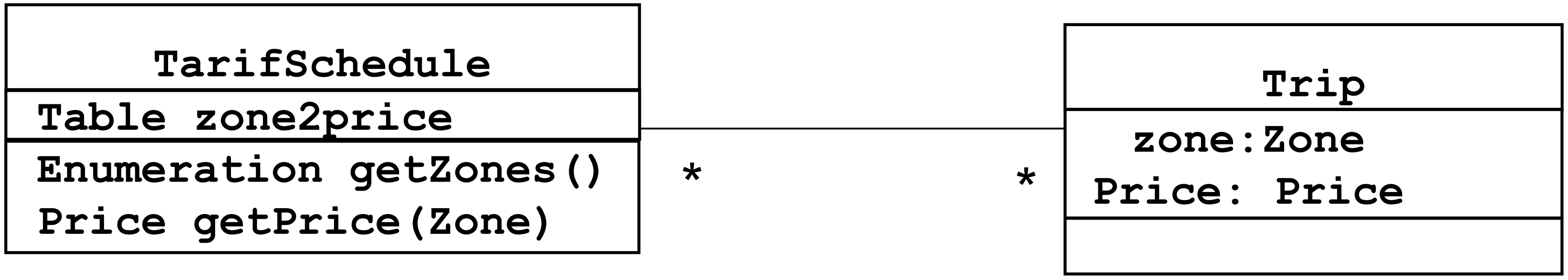
UML first pass: Use case diagrams



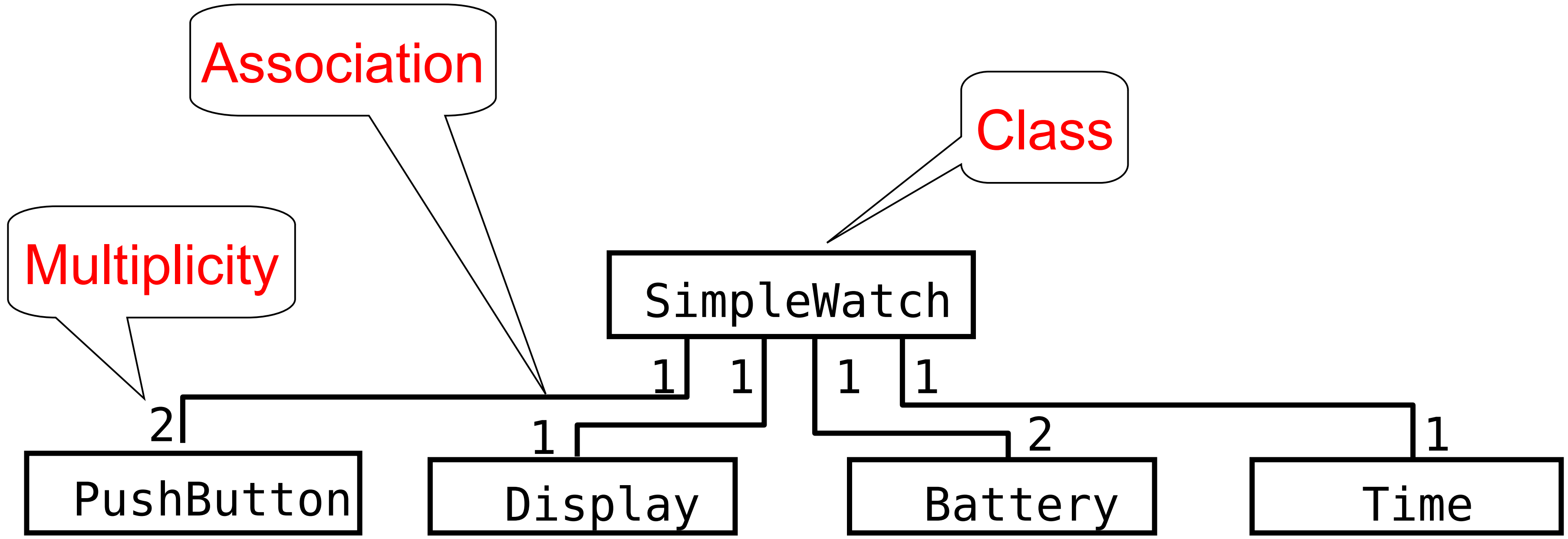
Use case diagrams represent the functionalities of the system from user's point of view

Class Diagrams

- Class diagrams represent the structure of the system
- Used
  - during requirements analysis to model application domain concepts
  - during system design to model subsystems
  - during object design to specify the detailed behavior and attributes of classes.



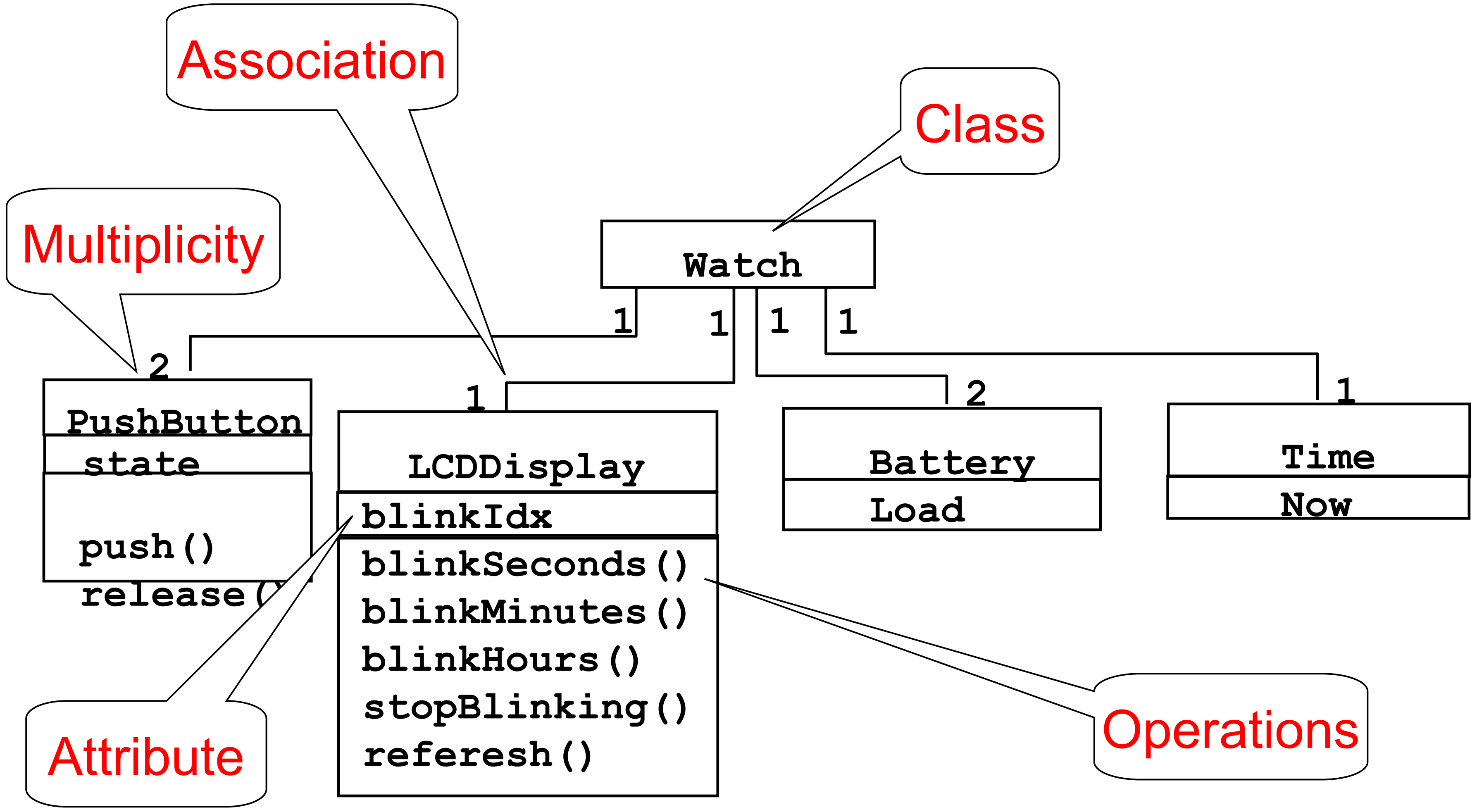
UML first pass: Class diagrams



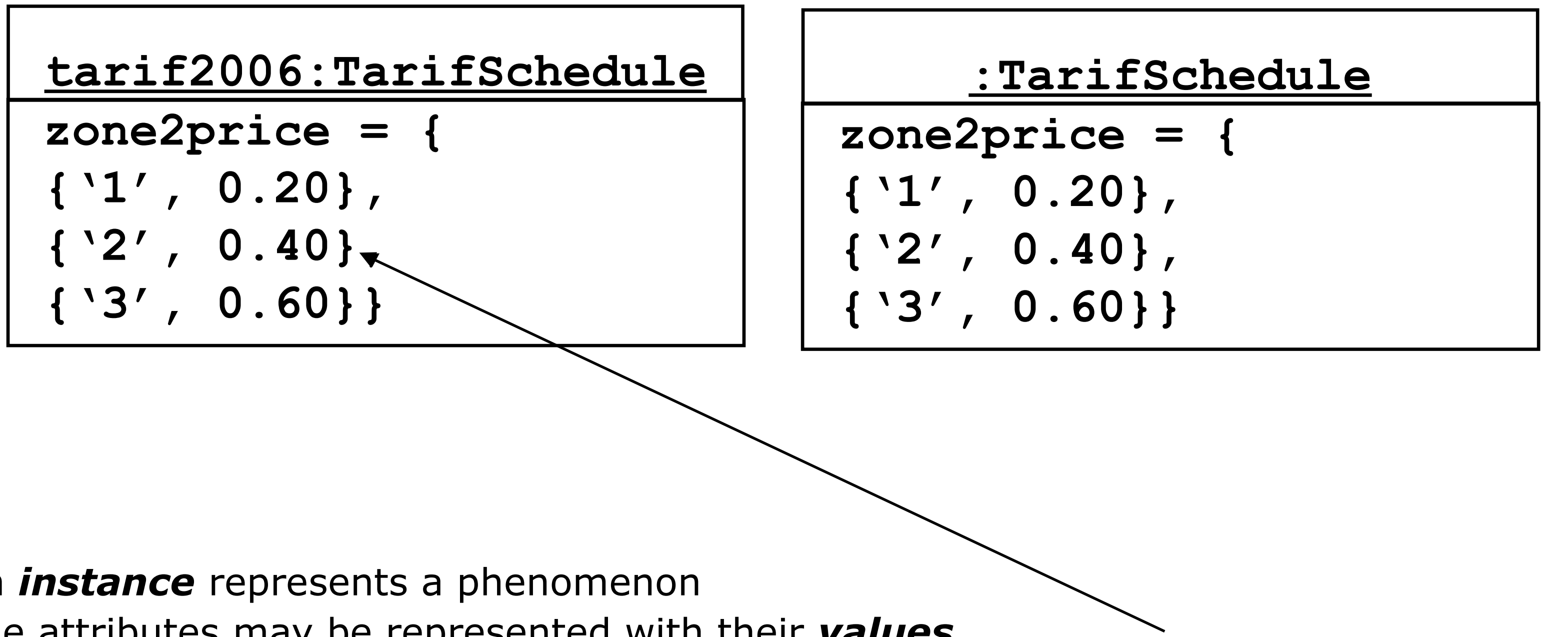
Class diagrams represent the structure of the system

UML first pass: Class diagrams

Class diagrams represent the structure of the system



Instances

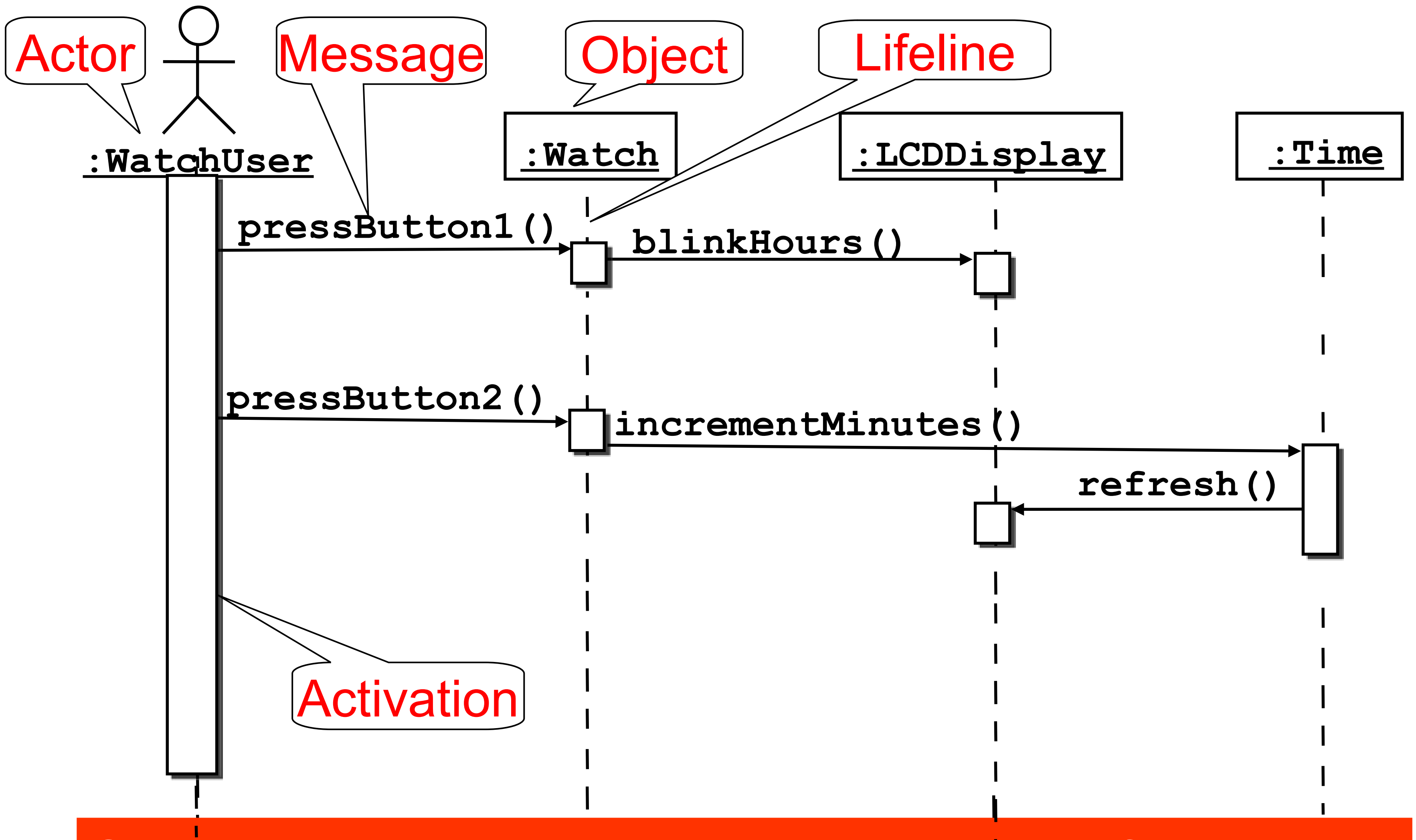


- An ***instance*** represents a phenomenon
- The attributes may be represented with their ***values***
- The name of an instance is <u>underlined</u>
- The name can contain only the class name of the instance (anonymous instance)

Actor vs Class vs Object

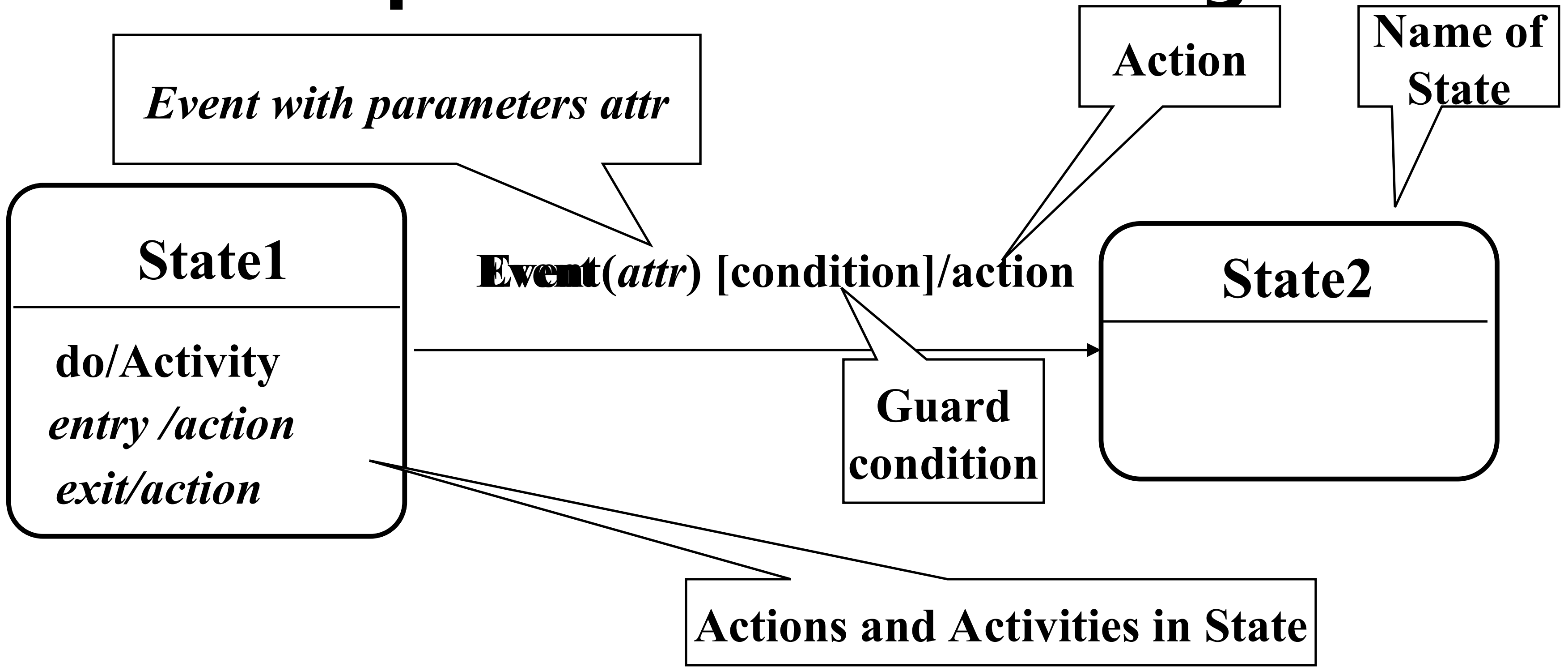
- **Actor**
  - An entity outside the system to be modeled, interacting with the system ("Passenger")
- **Class**
  - An abstraction modeling an entity in the application or solution domain
  - The class is part of the system model ("User", "Ticket distributor", "Server")
- **Object**
  - A specific instance of a class ("joe, the passenger who is purchasing a ticket from the ticket distributor").

UML first pass: Sequence diagram



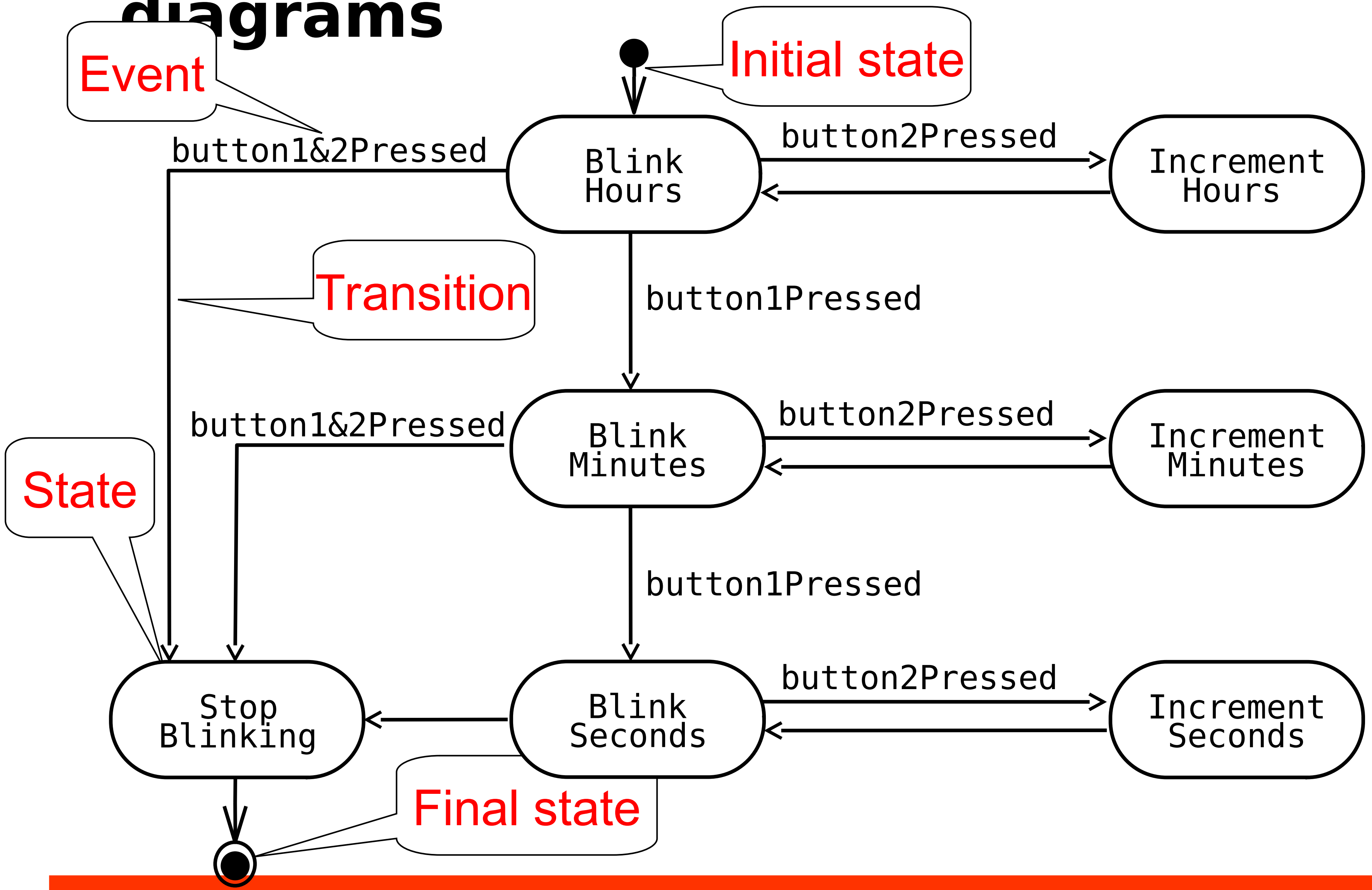
Sequence diagrams represent the behavior of a system as messages (“interactions”) between *different objects*

UML first pass: Statechart diagrams


Event with parameters attr
Action
Name of State
State1
do/Activity
entry /action
exit/action
Event(attr) [condition]/action
State2
Guard condition
Actions and Activities in State

- Note:
  - Conditions are enclosed with brackets: []
  - Actions and activities are prefixed with a slash /

UML first pass: Statechart diagrams



Represent behavior of *a single object* with interesting dynamic behavior.

State

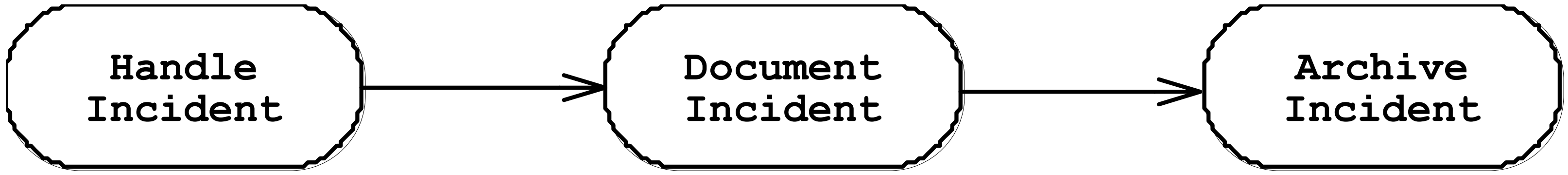
- An abstraction of the attributes of a class
    - State is the aggregation of several attributes of a class
- A state is an equivalence class of all those attribute values and links that do no need to be distinguished
    - Example: State of a bank
- State has duration

State Chart Diagram vs Sequence Diagram

- State chart diagrams help to identify:
  - Changes to an individual object over time
- Sequence diagrams help to identify:
  - The temporal relationship between objects over time
  - Sequence of operations as a response to one or more events.

Activity Diagrams

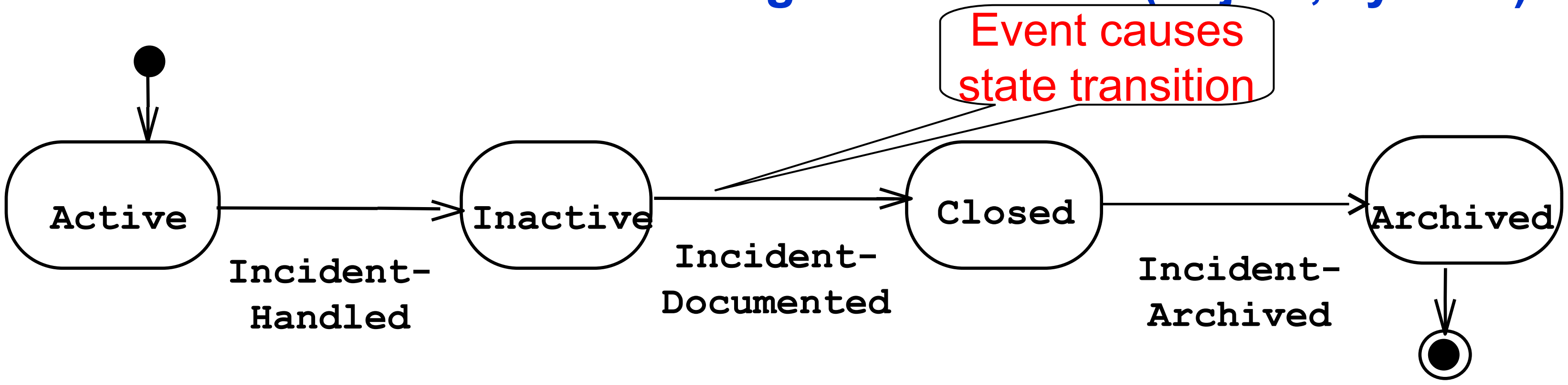
- An activity diagram is a special case of a state chart diagram
- The states are activities (“functions”)
- An activity diagram is useful to depict the workflow in a system



Activity Diagram vs. Statechart Diagram

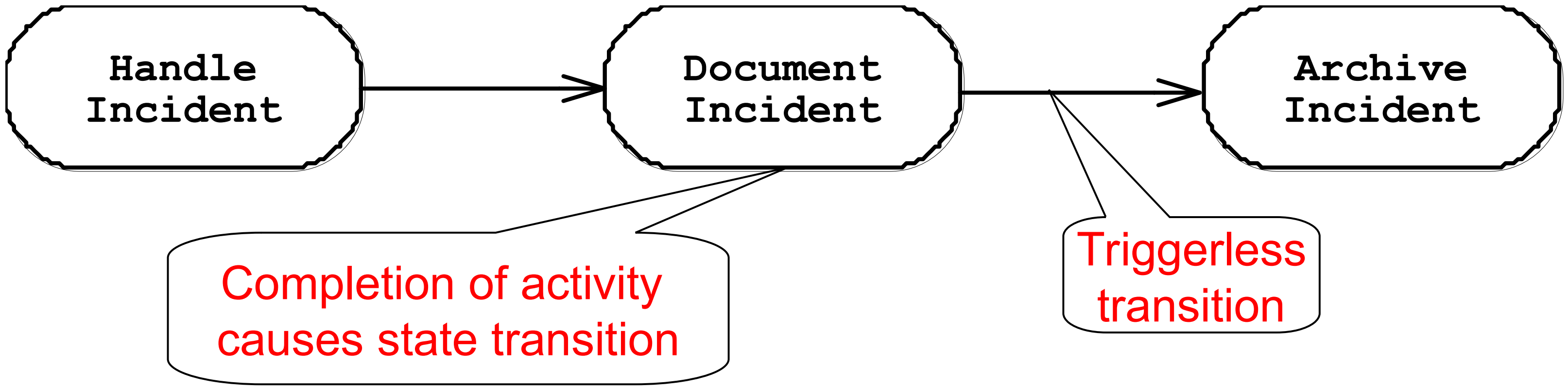
Statechart Diagram for Incident

Focus on the set of attributes of a single abstraction (object, system)



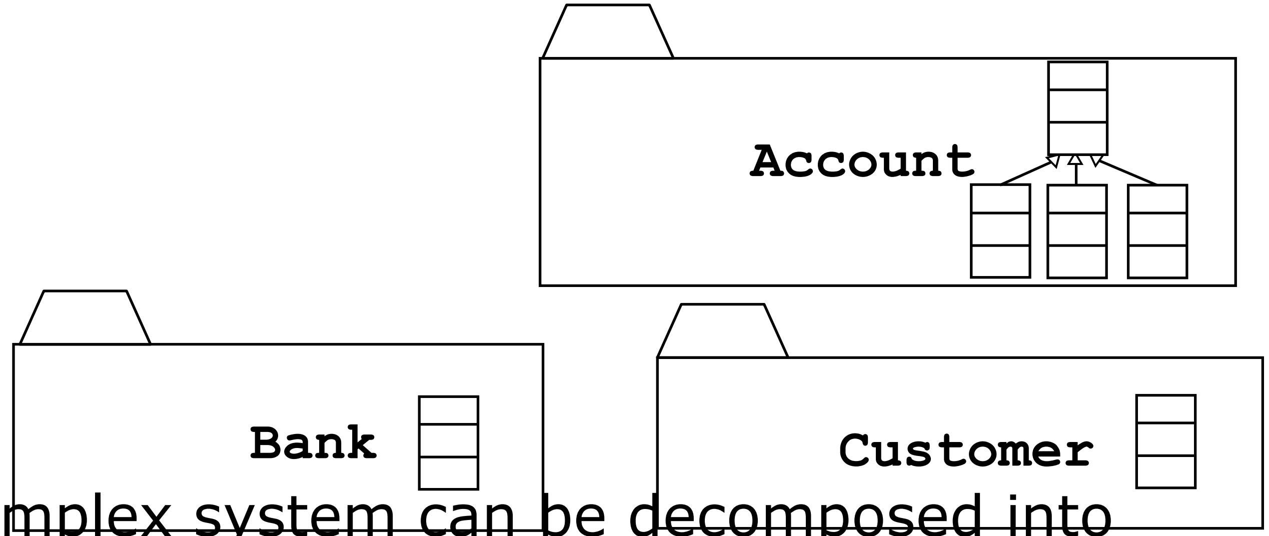
Activity Diagram for Incident

(Focus on dataflow in a system)



Packages

- Packages help you to organize UML models to increase their readability
- We can use the UML package mechanism to organize classes into subsystems



- Any complex system can be decomposed into subsystems, where each subsystem is modeled as a package.

Other UML Notations

UML provides many other notations

- We introduce them as we go along in the lectures
    - OCL: A language for constraining UML models.

What should be done first? Coding or Modeling?

- It depends….
- Forward Engineering
  - Creation of code from a model
  - Start with modeling
  - Greenfield projects
- Reverse Engineering
  - Creation of a model from existing code
  - Interface or reengineering projects
- Roundtrip Engineering
  - Move constantly between forward and reverse engineering
  - Reengineering projects
  - Useful when requirements, technology and schedule are changing frequently.

UML Summary

- UML provides a wide variety of notations for representing many aspects of software development
  - Powerful, but complex
- UML is a programming language
  - Can be misused to generate unreadable models
  - Can be misunderstood when using too many exotic features
- We concentrated on a few notations:
  - Functional model: Use case diagram
  - Object model: class diagram
  - Dynamic model: sequence diagrams, statechart and activity diagrams

Additional References

- Martin Fowler
  - UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed., Addison-Wesley, 2003
- Grady Booch,James Rumbaugh,Ivar Jacobson
  - The Unified Modeling Language User Guide, Addison Wesley, 2nd edition, 2005
- Open Source UML tools
  - http://java-source.net/open-source/uml-modeling