CMSC 132:
Object-Oriented Programming II

UML (Unified Modeling Language)

Department of Computer Science
University of Maryland, College Park

Overview

- Models & views
- Class diagrams
- Sequence diagrams
- UML class diagrams
- Use of class diagrams to represent relationships

UML

- **UML (Unified Modeling Language)**
  - Graphic modeling language for describing object-oriented software
  - Allows you to visualize the design of a program
  - Allow us to create an abstract model of a system
  - Started in 1994
  - Combined notations from 3 leading OO methods
    - OMT (James Rumbaugh)
    - OOSE (Ivar Jacobson)
    - Booch (Grady Booch)

UML

- Industry standard
- Many features
  - Large collection of notations
  - Multiple views
  - Multiple diagrams
- We focus mainly on
  - Logical view of relationship between classes

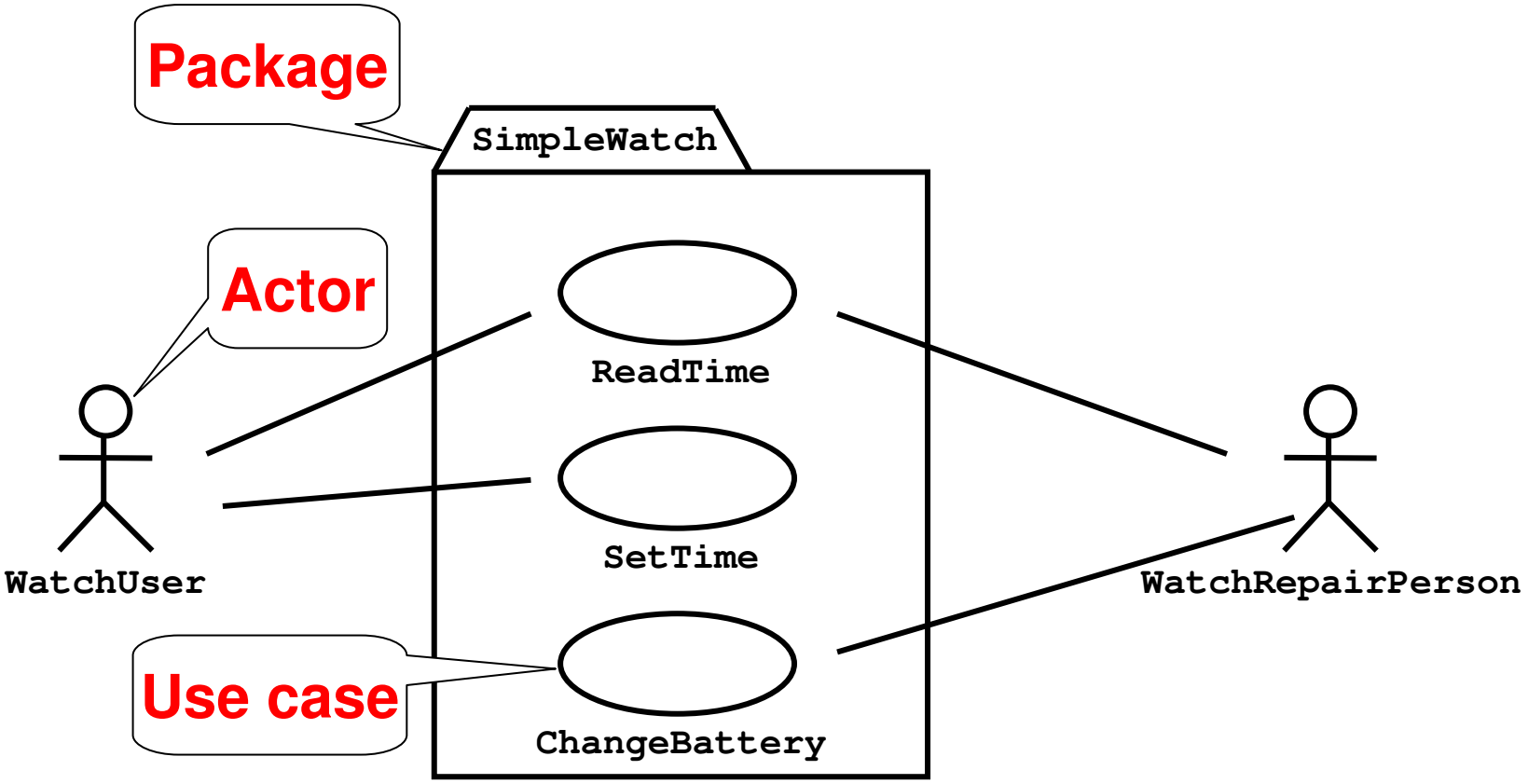
UML Motivation

- Software growing larger & complex
  - Difficult to analyze
- Need to describe software design
  - Clearly
  - Concisely
  - Correctly
- UML equivalent to software “blueprint”
  - Provides simple yet clear abstraction for software

(Some) UML Diagrams

- **Class**
  - Describe static structure of the classes in system
- **Use Case**
  - Each use case provides one or more *scenarios* that convey how the system should interact with the end user or another system to achieve a specific business goal (from Wikipedia)
  - Use cases typically avoid technical jargon, preferring instead the language of the end user or domain expert (from Wikipedia)

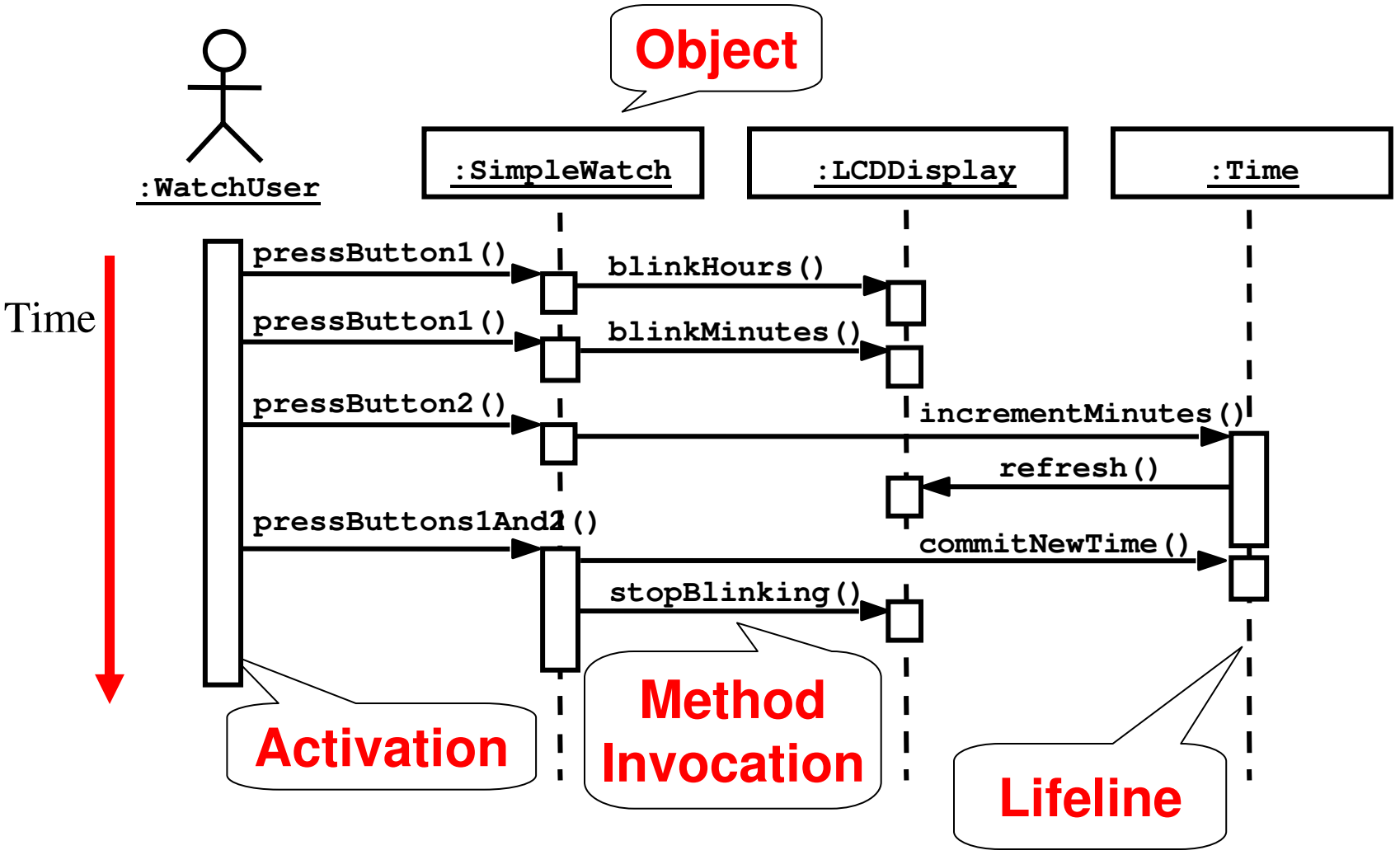
Use Case Diagrams



(Some) UML Diagrams

- **Sequence**
  - **Describes interactions among objects by presenting the order of method invocations.**
  - **Usually developed on a use case basis showing the message sequence for a particular use case**

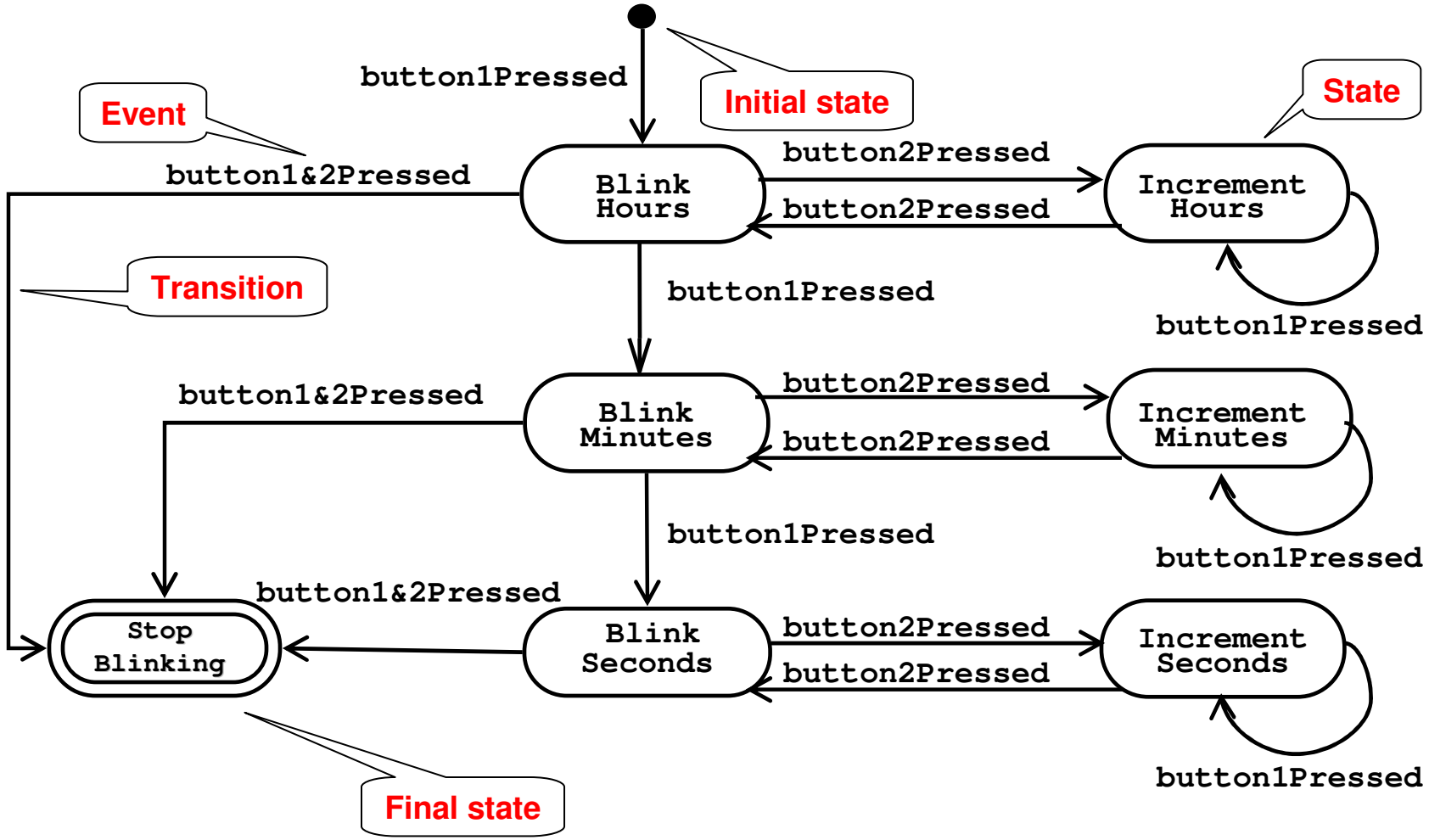
Sequence Diagram



Sequence Diagrams show the message sequence

State Diagrams

Describes dynamic behavior of objects as finite state machine

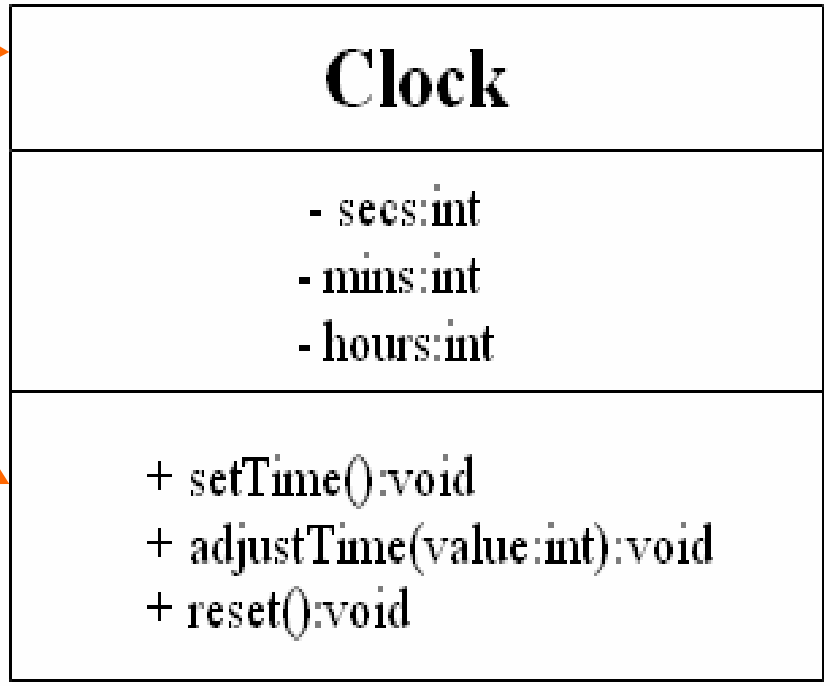


UML Class Diagrams

- Represent the (static) structure of the system
- During analysis
  - Used to model problem domain concepts
- During detailed design
  - Used to model classes
- UML class diagrams allow us to depict different kinds of relationships among the classes

Class Diagrams

- Class Representation
  - Name
  - State
  - Behavior
  - Visibility
    - + public
    - - private
    - # protected
    - ~ package



  - Attribute Type – Specified following the name and separated by colon
  - Static fields/methods are underlined

UML Class Diagrams $\leftrightarrow$ Java Code

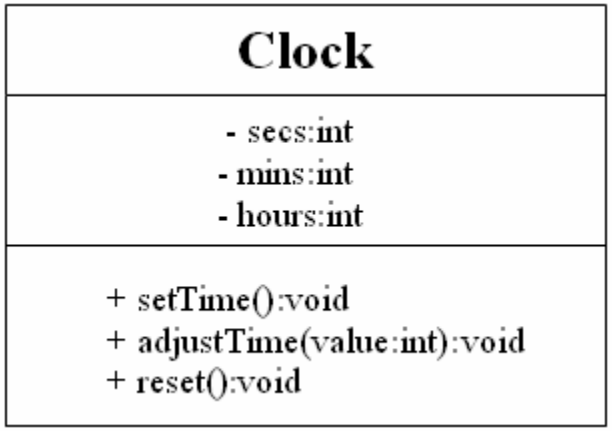
- Different representation of same information
  - Name, state, behavior of class
  - Relationship(s) between classes
  - Should be able to derive one from the other
- Motivation
  - UML $\Rightarrow$ Java
    - Implement code based on design written in UML
  - Java $\Rightarrow$ UML
    - Create UML to document design of existing code

Java → UML : Clock Example

- **Java**

```
class Clock { // name
    // state
    private int seconds;
    private int minutes;
    private int hours;
    // behavior
    public void setTime();
    public void adjustTime(int value);
    public void reset();
}
```

Java Code



Class Diagram

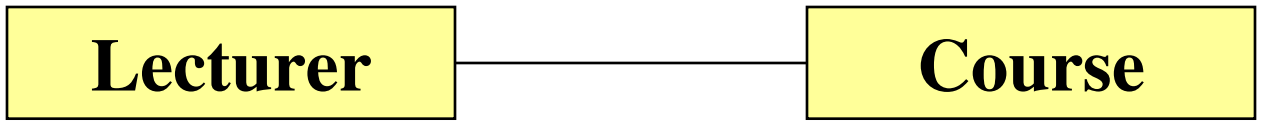
Relationships Among Classes

- UML class diagrams allow us to depict different kinds of relationships among the classes.
- Types of relationships
  - Association
  - Generalization
  - Dependency

Relationships Among Classes

Association

- Binary relationship describing an activity between objects of two classes
- Example: Lecturer teaching a course is an association between a Lecturer class and a Course class

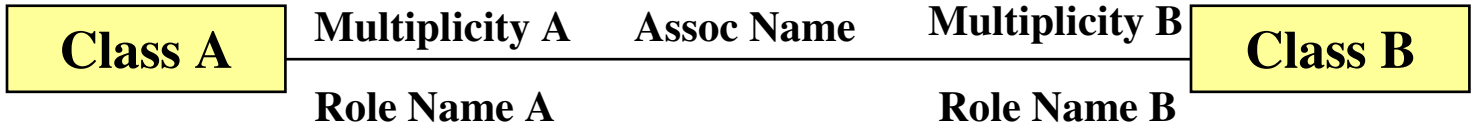


- An association is represented by a class instance variable (data field) or by a parameter

Relationships Among Classes

Association

- UML Notation for Association



- An association is represented by a class instance variable (data field) or by a parameter
- Multiplicity X – number of objects of class X associated with the other class. Multiplicity can be a number or range of values.
- Navigation Direction – Arrows can be placed at one or both ends of the line. An arrow from class X to class Y implies objects of class X can send messages to objects of class Y. No arrows may represent navigation in both directions is possible or no navigation information is being shown

Relationships Among Classes

Association

- UML Association Examples