



## Lecture 5: Modeling Software Behaviour

- UML sequence Diagrams
- Comparing Designs
- Explaining Design Patterns
- Style tips



## Things to Model

### E.g. Structure of the code

- Code Dependencies
- Components and couplings

### E.g. Behaviour of the code

- Execution traces
- State machines models of complex objects

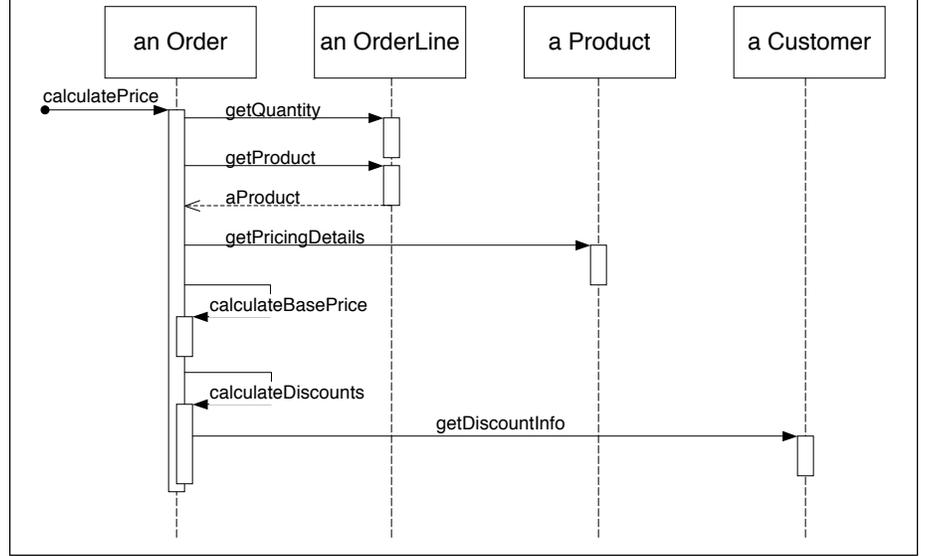
### E.g. Function of the code

- What functions does it provide to the user?

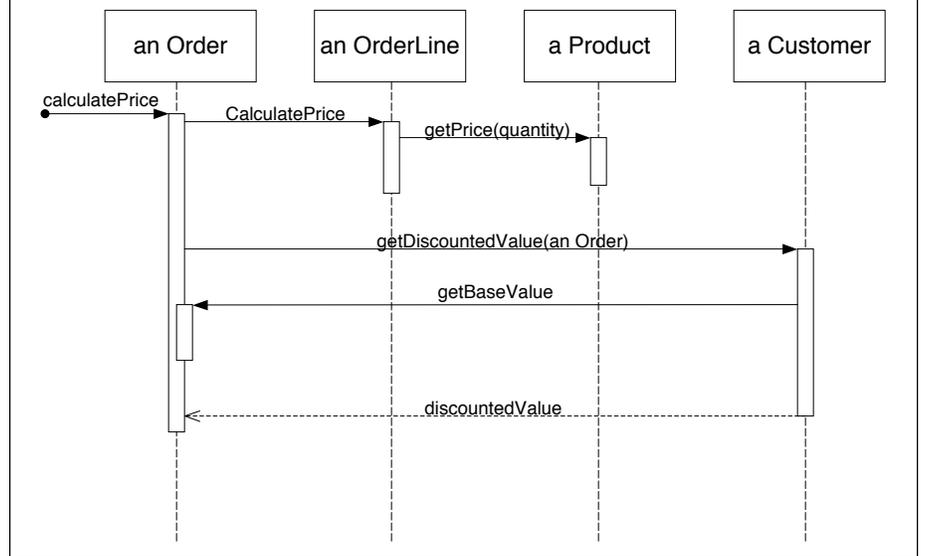




# Sequence Diagrams

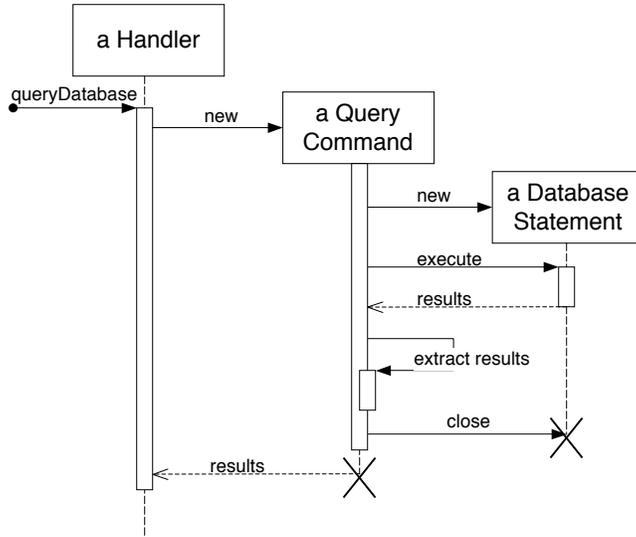


# Design Choices...

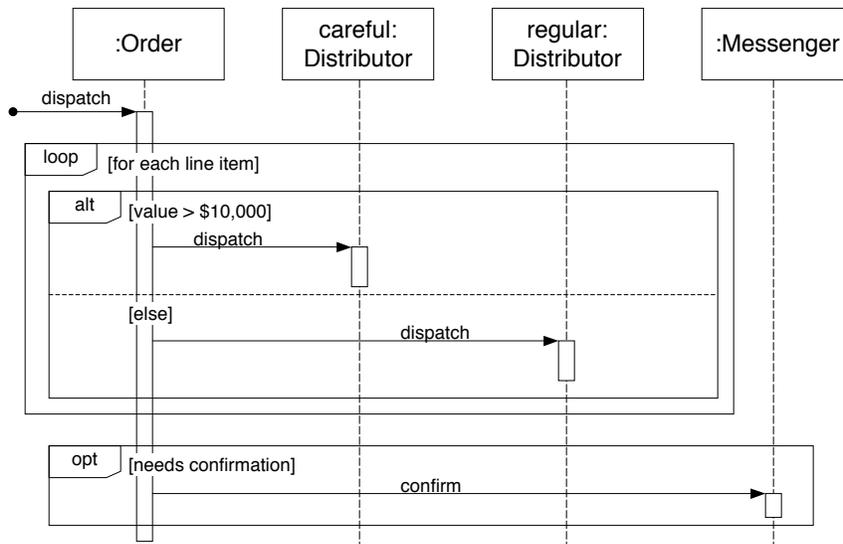




# Creating and Deleting Objects



# Interaction Frames





# When to use Sequence Diagrams

## Comparing Design Options

Shows how objects collaborate to carry out a task  
Graphical form shows alternative behaviours

## Assessing Bottlenecks

E.g. an object through which many messages pass

## Explaining Design Patterns

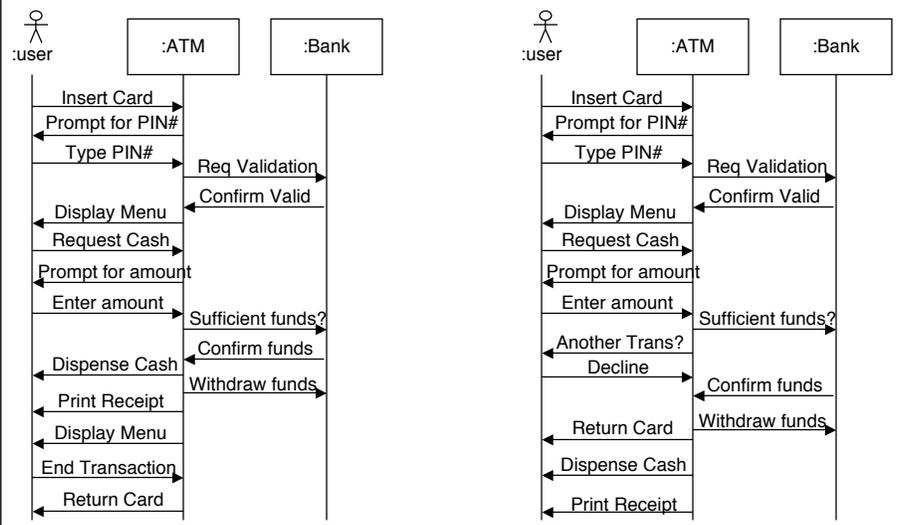
Enhances structural models  
Good for documenting behaviour of design features

## Elaborating Use Cases

Shows how the user expects to interact with the system  
Shows how the user interface operates



# Comparing Designs

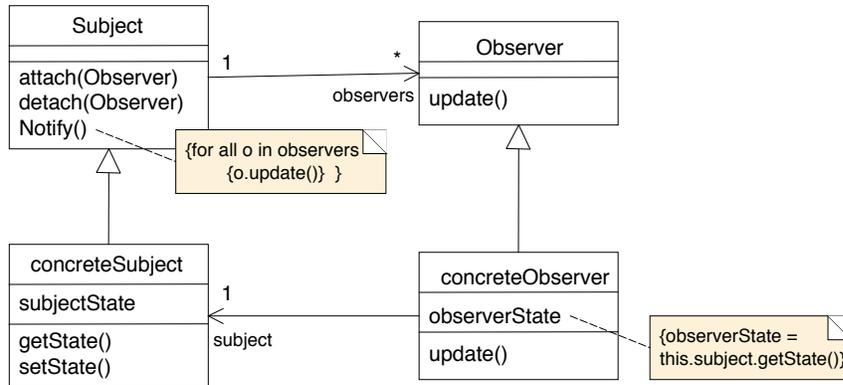




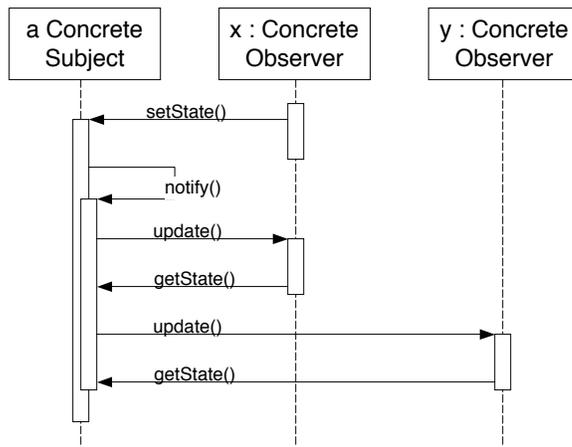
# Modeling a Design Pattern

## E.g. Observer Pattern

For a one-to-many dependency, when you need to maintain consistency  
The subject pushes updates to all the observers



# Sequence Diagram for Observer





# Style Guide

## Spatial Layout

- Strive for left-to-right ordering of messages
- Put proactive actors on the left
- Put reactive actors on the right

## Readability

- Keep diagrams simple
- Don't show obvious return values
- Don't show object destruction

## Usage

- Focus on critical interactions only

## Consistency

- Class names must be consistent with class diagram
- Message routes must be consistent with (navigable) class associations

