

# **1.264 Lecture 5**

## **Unified Modeling Language (UML)**

# Unified Modeling Language

- **Graphical, object-oriented modeling language. Options:**
  - Sketch language to define system requirements
  - Blueprint language for system design
  - Implementation language to automatically generate software
- **‘Open standard’ managed by Object Management Group**
  - Many implementations of UML (Microsoft, IBM, Borland, ...)
- **Why is UML coming into wide use?**
  - Speeds up requirements process
  - Lessens information loss between requirements and design processes, and between design and implementation
  - Clearer than natural language
    - Provides a level of precision, but avoids details
  - Supports iterative development (i.e., spiral model)
    - Supports both high level requirements/design in early spirals and detailed requirements/design later
  - Hope that analysts can produce software without programmers

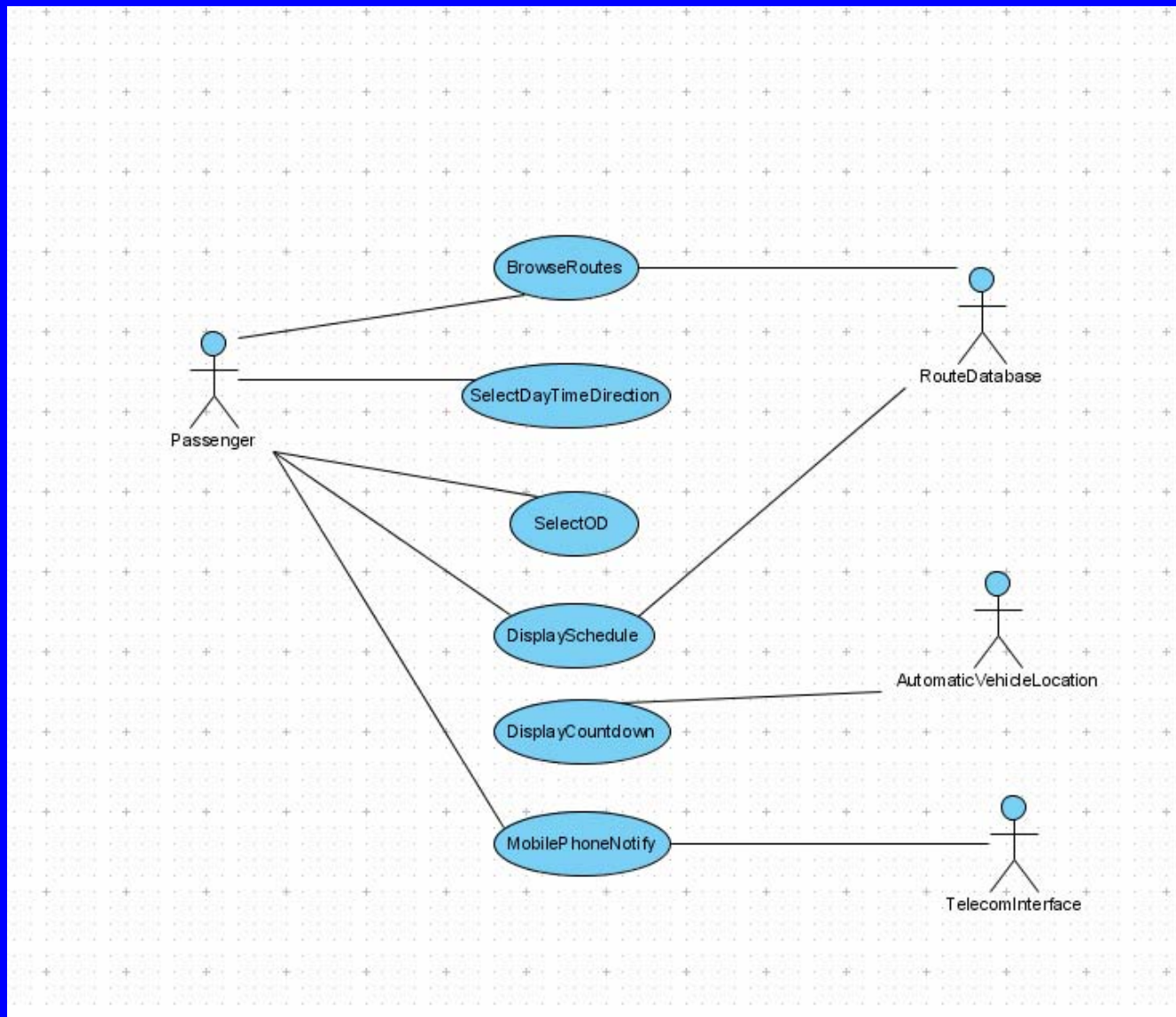
# Unified Modeling Language uses

- **Requirements:**
  - Use cases, which are very structured scenarios used to define system requirements
    - Good basic approach, but needs narrative to support
  - Class diagrams
    - Show relationships and behavior of all objects (things) in system
    - Derived from data models, which we cover in next lecture
  - Component diagrams to show high level view of system
- **Design:**
  - More detailed use cases, class diagrams, component diagrams
  - Activity and/or sequence diagrams, used to model workflows, to find related or duplicate processes that can be generalized
  - State diagrams for complex objects
  - Other diagram types, as needed
- **Implementation:**
  - Class, state and other diagrams (vendor-specific)

# Use cases

- **Capture requirements of system as structured scenarios**
  - Use case diagrams capture how use cases relate to each other
  - The actual use cases are usually text
  - A note: Users are called 'actors'
- **Exercise:**
  - Passenger browses bus routes and selects one to get info
  - Passenger selects day of week, time period and direction
  - Passenger selects origin and destination points
  - System displays schedule and countdown clock to next bus
  - Passenger opts for mobile phone notification of given bus at given stop
- **What alternatives are there?**
  - Passenger selects origin, destination first
  - Passenger clicks on map
  - Etc.
- **What can go wrong with the chosen alternative?**
  - No service at requested time or stops; out of town phone,...

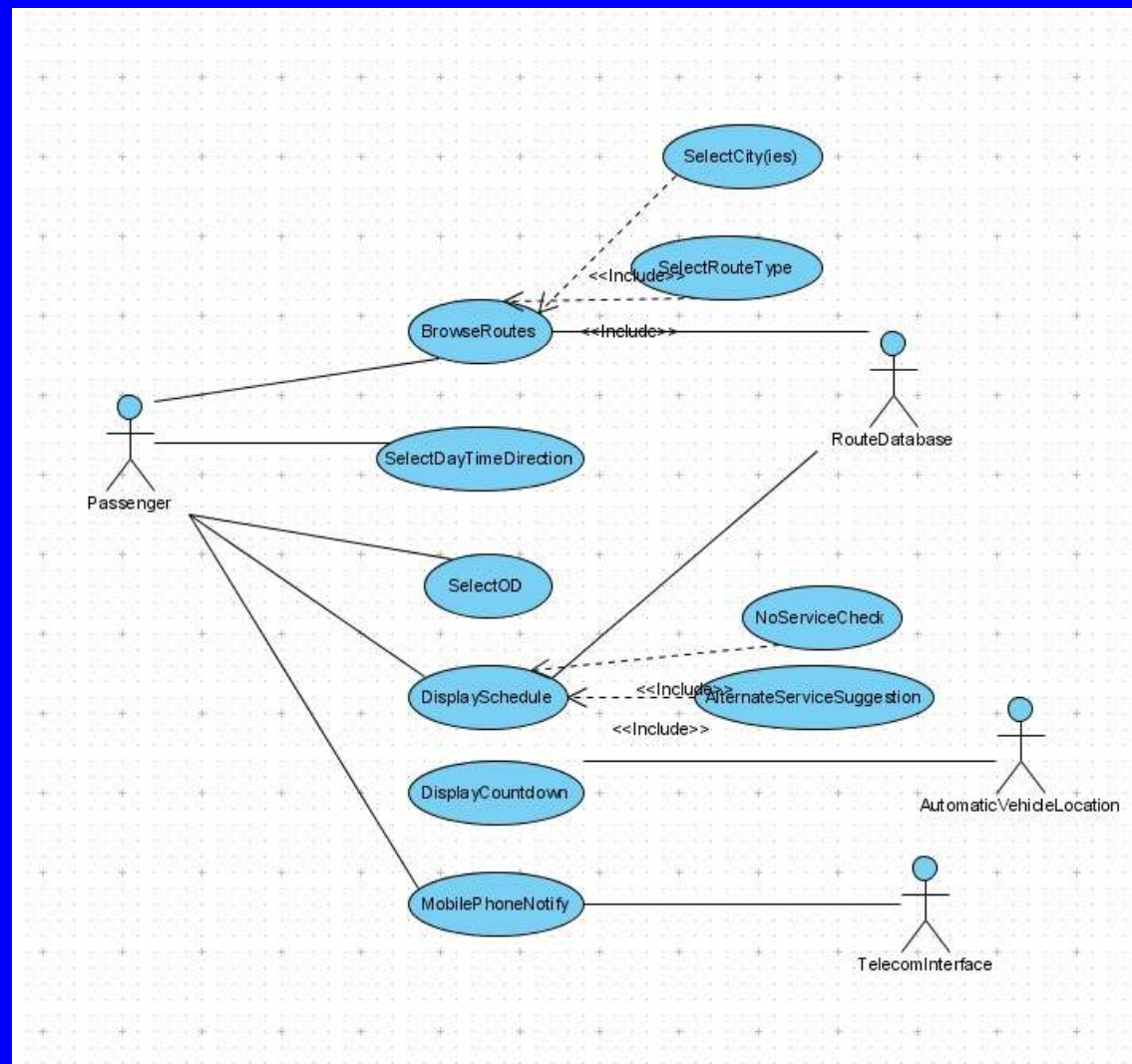
# Use case example



## Use case exercise

- Add one of the exception cases:
  - No service
  - Out of town phone
  - Others...
- Create 'included' use cases for one diagram case:
  - Break down complex use cases into smaller one
  - (You can use 'extend' if you want, but it's ill-defined)

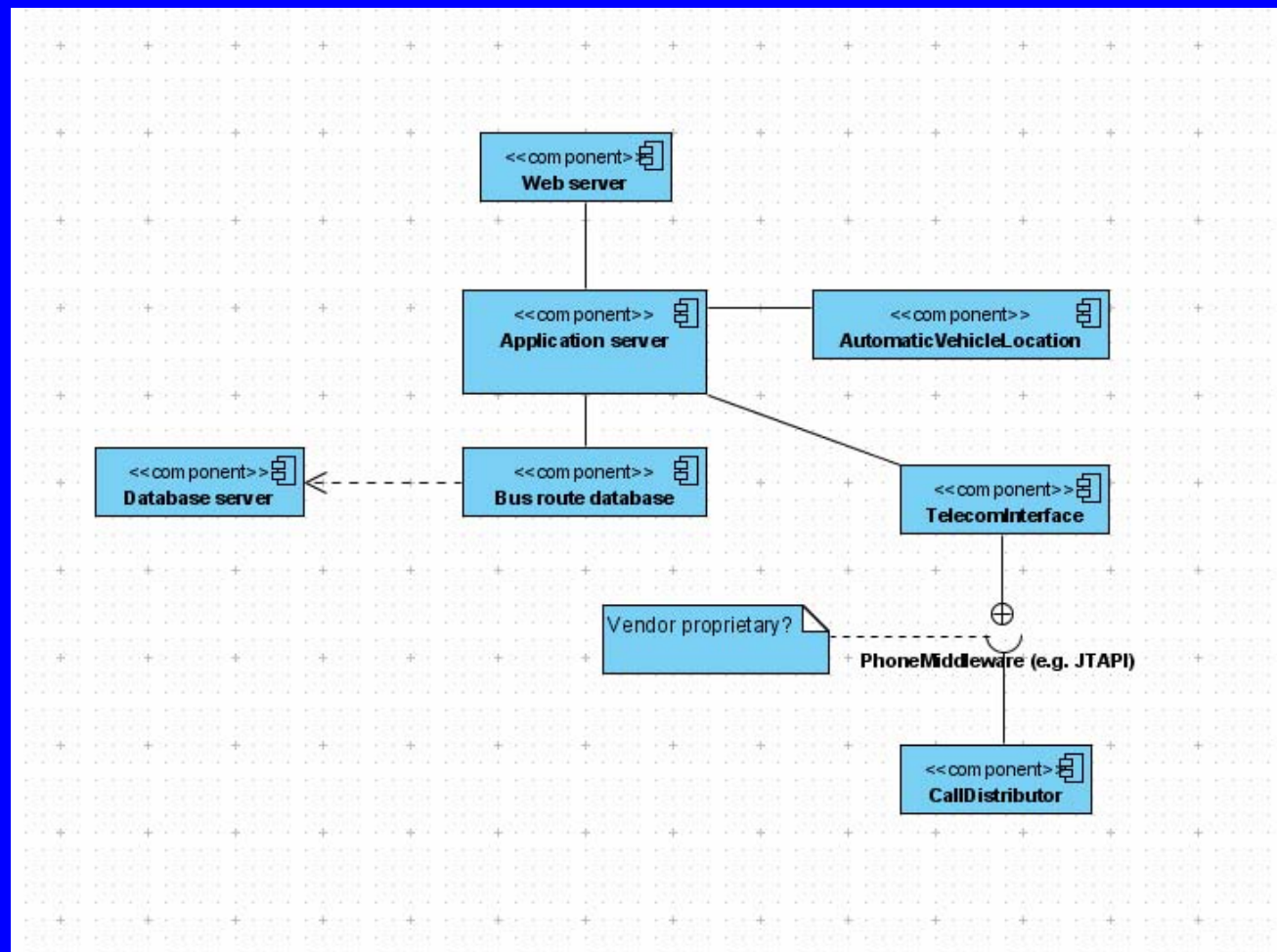
# Use case solution



## Component diagram

- **Draw the components for the bus information system example:**
  - **Web server**
  - **Application server (creates Web pages dynamically)**
  - **Database server (holds bus route database)**
  - **Telecom system interface from Web server**
  - **Telecom system**

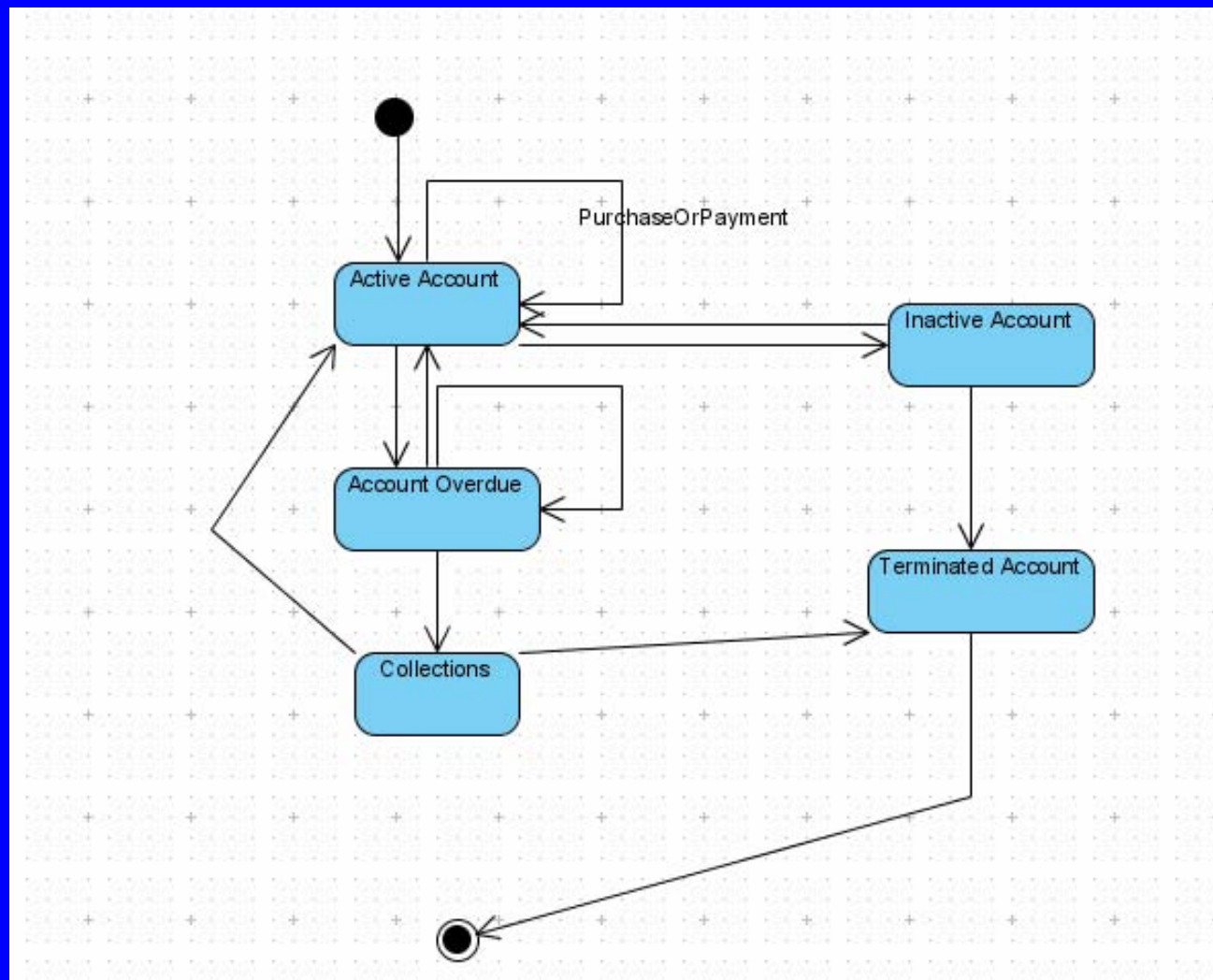
# Component diagram example



# Dynamic UML models

- **While static models (use cases, class diagram, component diagram) are done for the system as a whole, dynamic models are done only for key components**
- **State diagram**
  - Specifies behavior of a single object
- **Sequence diagram**
  - Shows details of one scenario and messages that flow between objects in that scenario over time
  - Heavily used in standards
- **Activity diagram**
  - Shows flow of logic, data, messages
  - Often less structured than other UML diagrams
  - Replaces flow charts
- **Communication diagram**
  - Shows flow of messages as a graph
  - Used as variant of sequence diagram
- **Others, as needed**

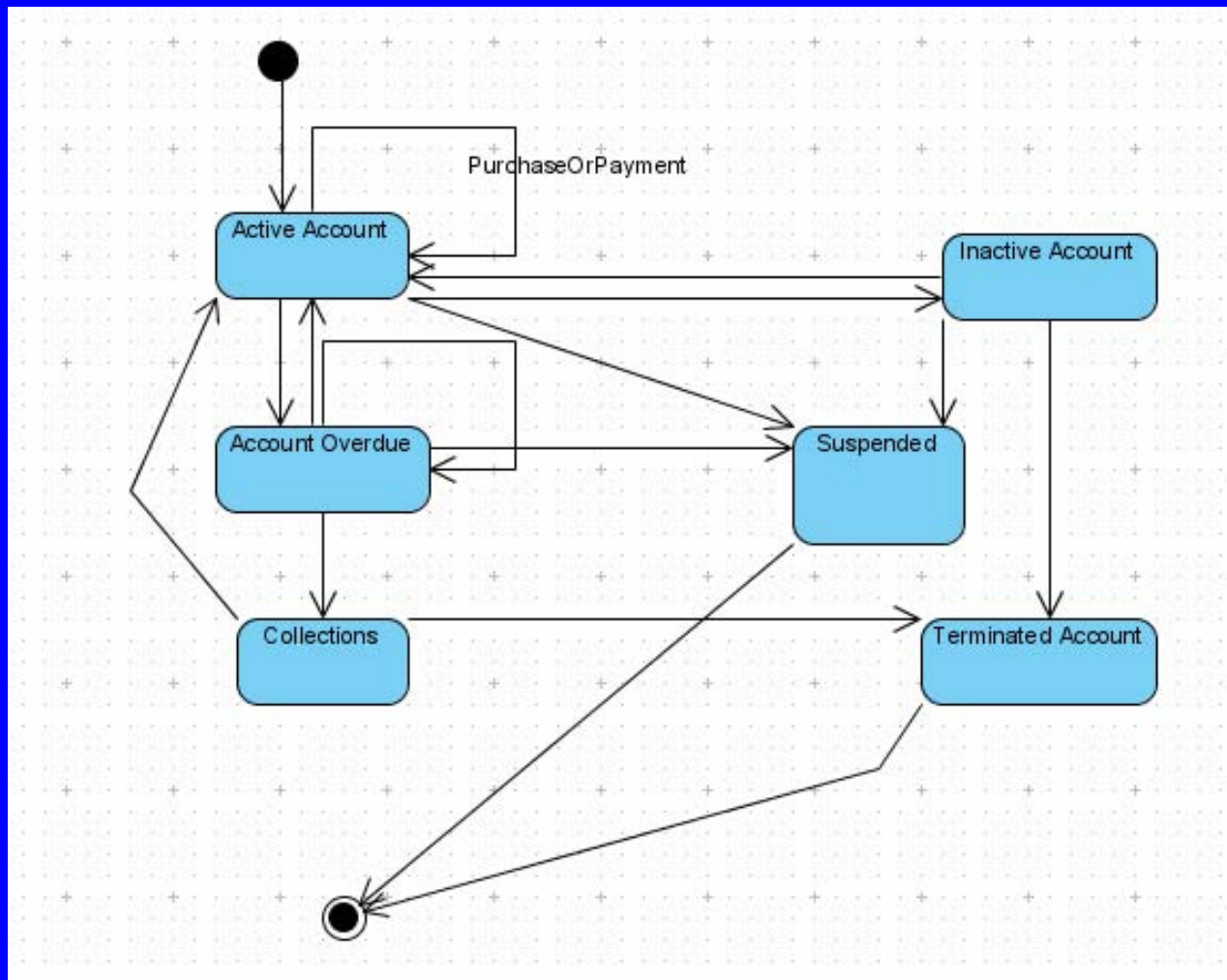
## State diagram example



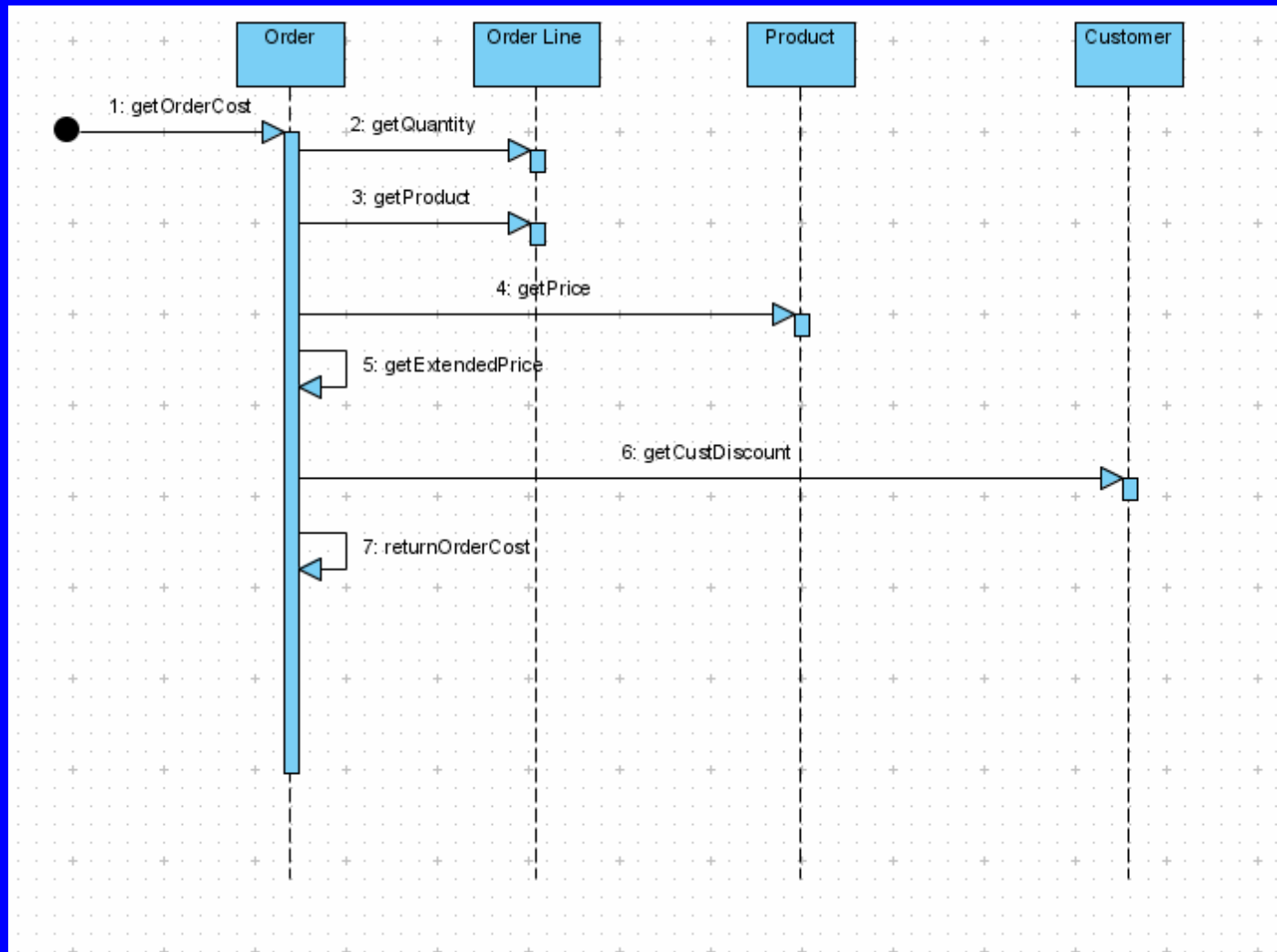
## State diagram exercise

- **Model one other state:**
  - Suspended account (identity theft)

# State diagram solution



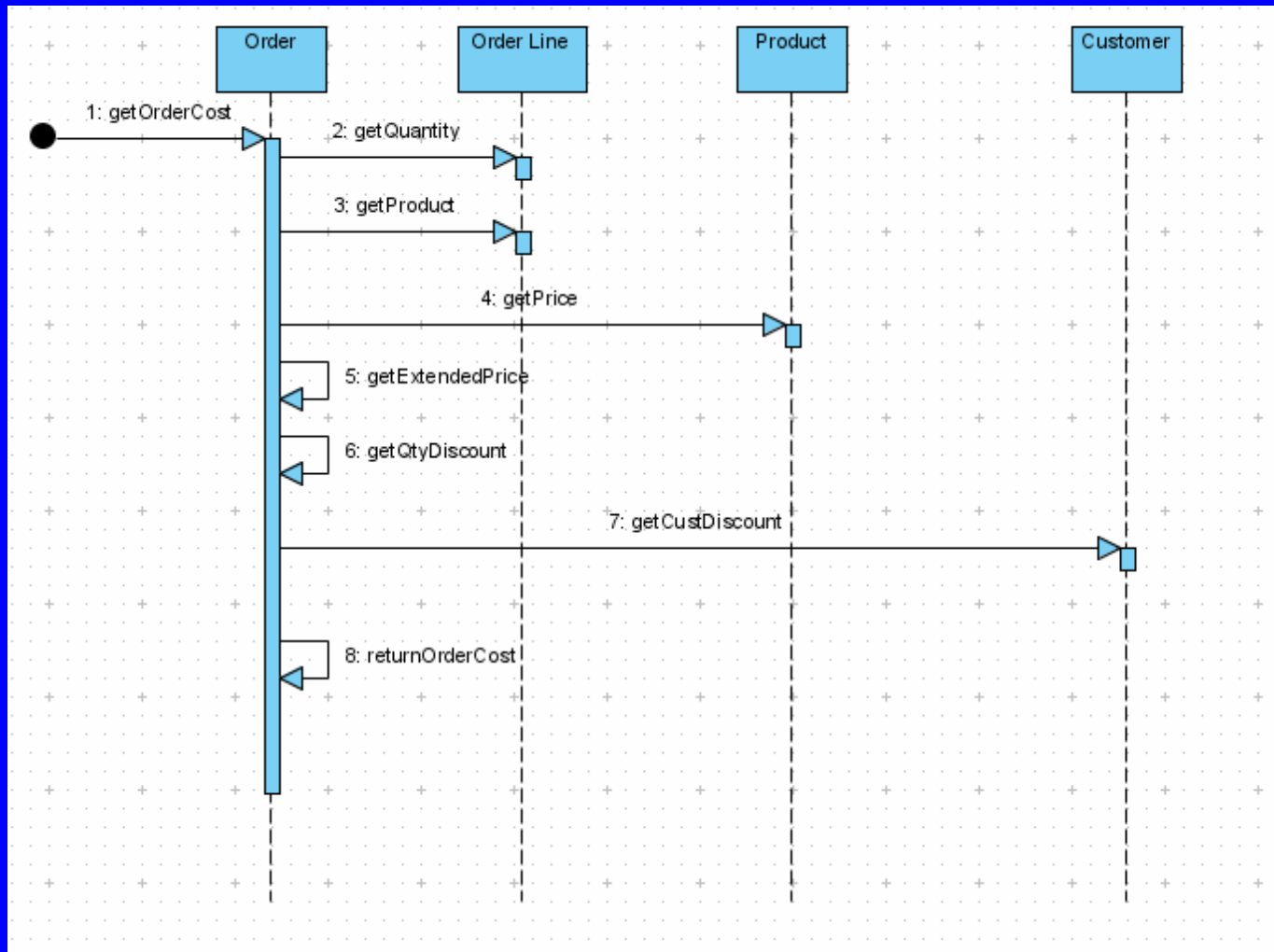
# Sequence diagram example



## Sequence diagram exercise

- **Add a discount calculation based on the total order quantity, in addition to the customer discount**

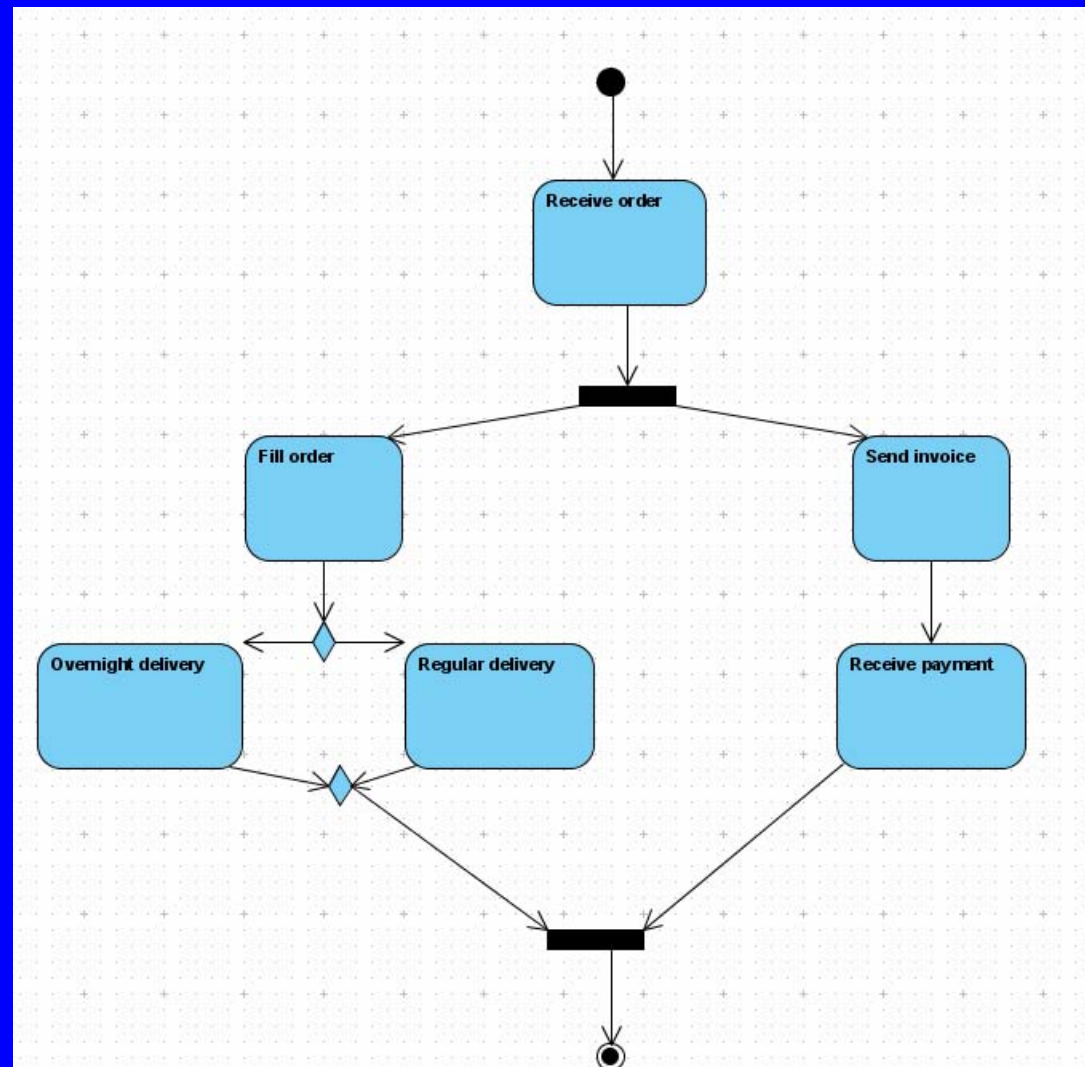
# Sequence diagram solution



## Activity diagram

- Shows flow of messages, logic, actions
- This is at a much higher level of abstraction than flow charts
  - Flow charts show logic for single method (if statements, loops, etc.)
  - Activity diagrams show flow among objects

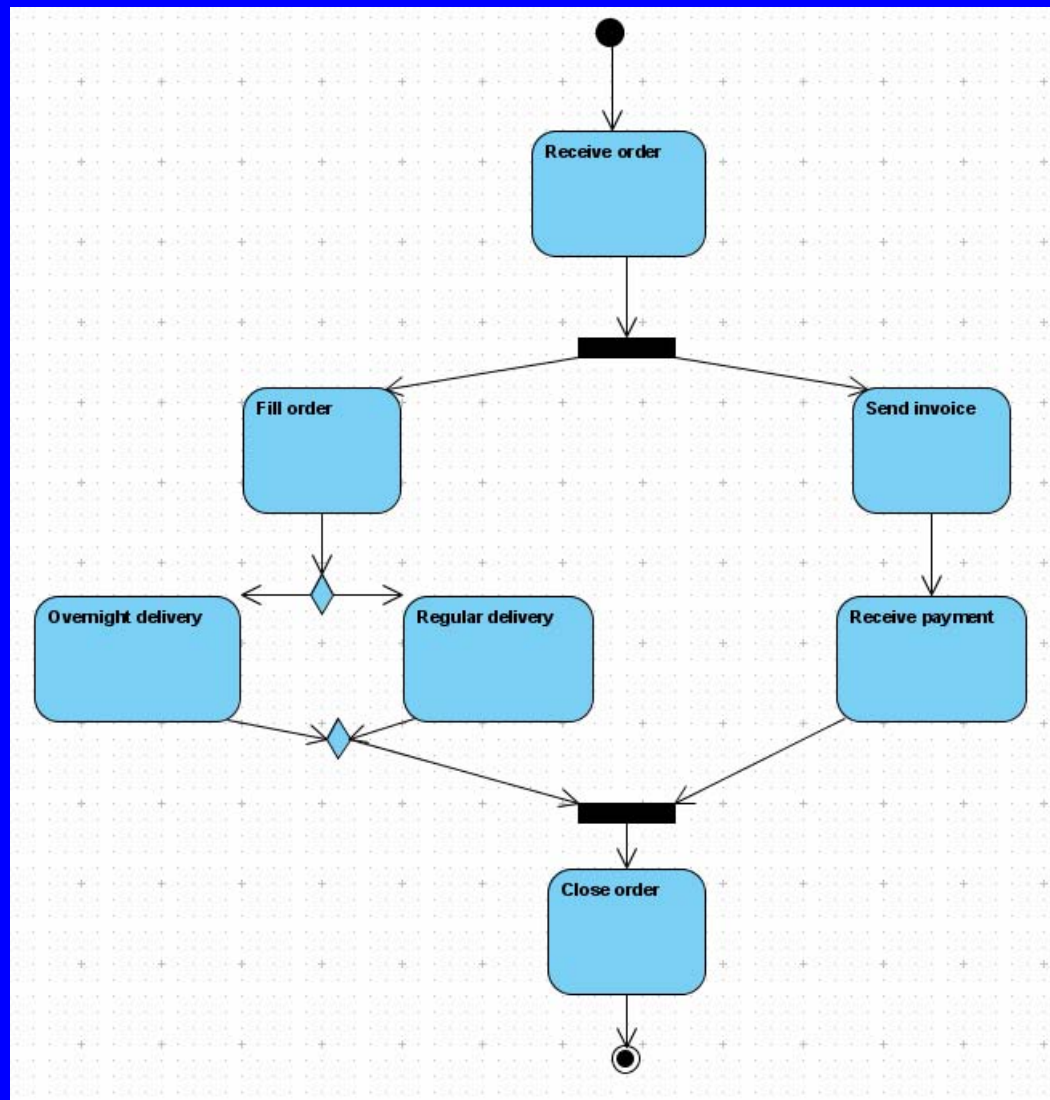
# Activity diagram example



## Activity diagram exercise

- Add a 'close order' activity

# Activity diagram solution



# UML Summary

- Use UML after writing scenarios and narratives as an initial requirements document
  - Refine them into use cases
- Prepare the initial data model (next lecture)
  - Add operations/methods to the entities, after understanding the data, to create a class diagram
- Use UML component diagrams to give overview of the system, in requirements
- Use UML state diagrams, sequence diagrams and activity diagrams to specify objects and processes
  - Prepare these selectively for complex or interesting objects
- UML is becoming a 'universal' language: new staff coming to a project can read it, and this reduces the learning curve very substantially
  - Developers and analysts can both understand it readily
  - I use UML even for analysis-only projects (as well as writing requirements and modeling data)