UML
-
Diagram Types

Overview

1. Class diagram
2. Activity diagram
3. Use case diagram

Class diagram - General

- Class
- Activity
- Use case

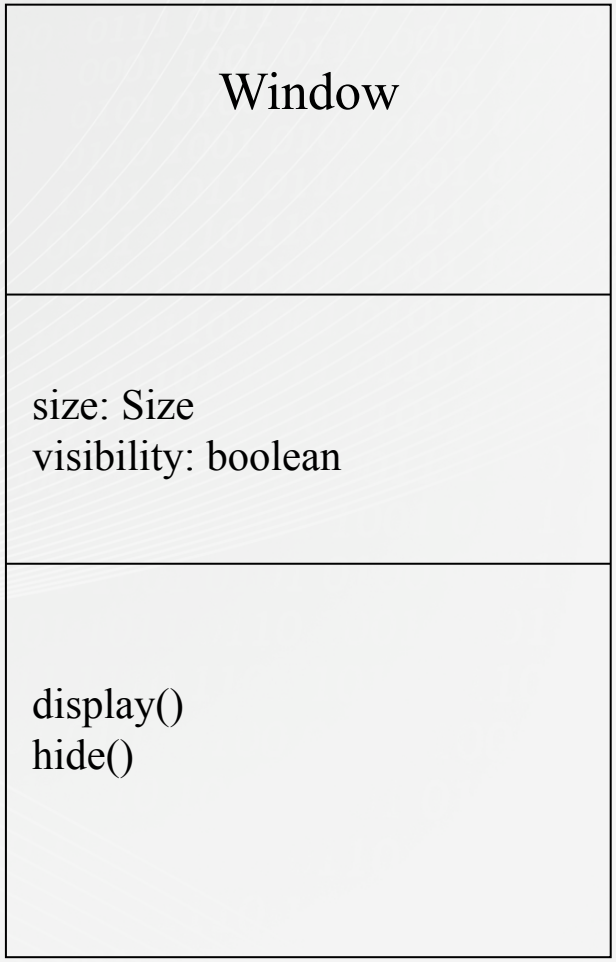
- Diagram describng the structure of a system

- Shows attributes:
  - Classes
  - Methodes
  - Realtionships

Class diagram - Notation

- Class
- Activity
- Use case

- Name
- Attributes
- Methods
- + → Public
- - → Private
- # → Protected



Class diagram - Notation

- Class
- Activity
- Use case

- Assoziation

- Employee works at a company
  - Numbers can define several new information of an association

Class diagram - Notation

- Class
- Activity
- Use case

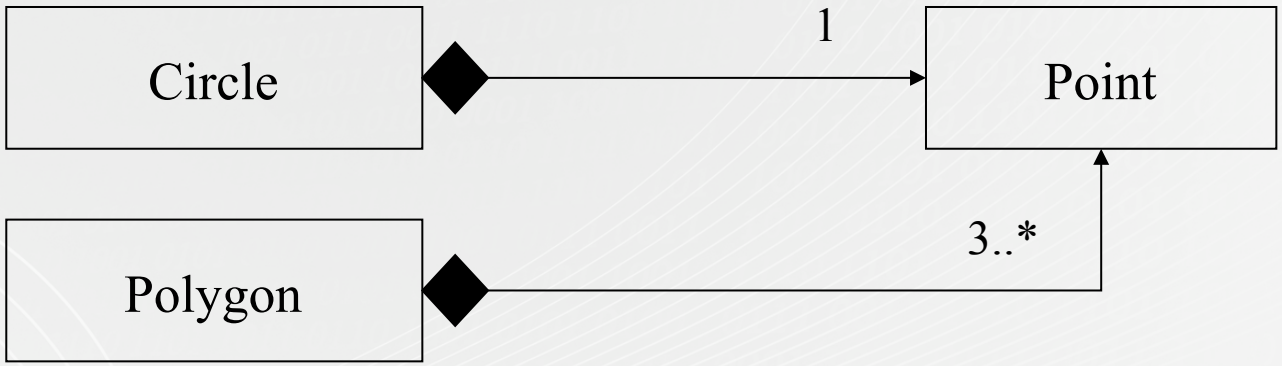
- Aggregation

- The door is a part of a car or a house

Class diagram - Notation

- Class
- Activity
- Use case

- Composition



- The life time of the part is dependent upon the whole

→ If there is no polygon there is no point

Class diagram - Notation

- Class
- Activity
- Use case

- Dependency

- Weak relationship

→ It just uses at some point one function

- Generalization

→ Relation between classes

→ Abstract class

Class diagram – Ad-/Disadvantages

- Class
- Activity
- Use case

- <u>Advantages:</u>
- Forces the programmer to think out the structure of classes before writing code

  → Lead to a more robust application
- Blueprint for maintenance programmers

  → Get an overview of the structure
- <u>Disadvantages:</u>
- Programmer needs to learn uml to write an propercalss diagramm
- Too much time is spent on designing a class diagramm
- Whole code development more complicated.

- ✓ Class
- ➢ Activity
- ● Use case

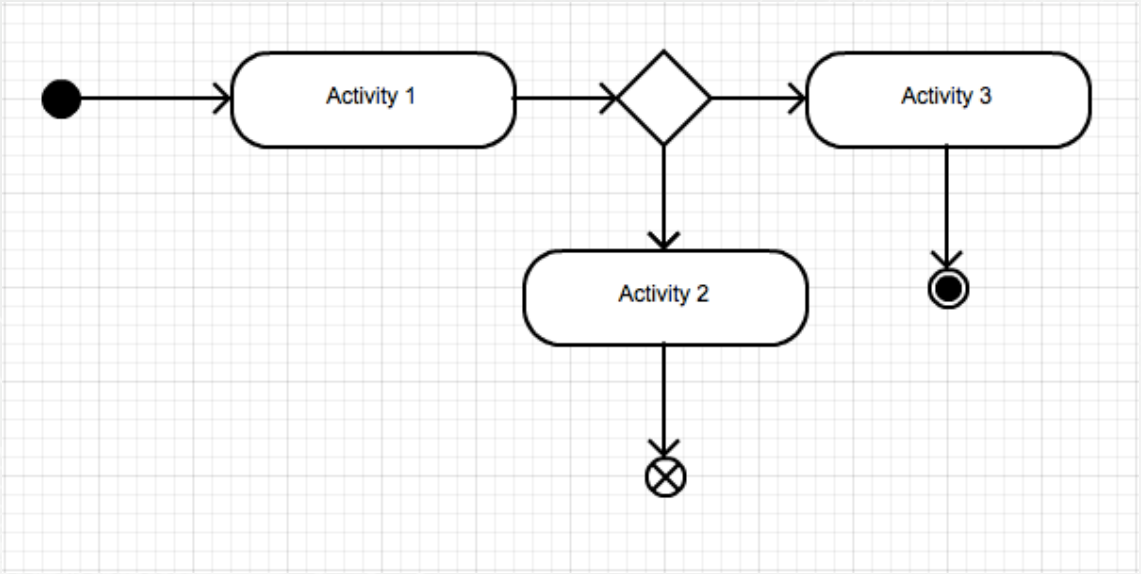
Activity diagram - General

- Graphical representations of workflows
- Limited number of shapes, connected with arrows
- Shapes
  - Rounded rectangles represent actions
  - Diamonds represent decisions
  - Bars represent fork and joins
  - Circles represent start and end
  - Arrows represent the flow

Activity diagram - Notation

- ✓ Class
- ➢ Activity
- ● Use case

- Start/stop



- Solid circle - beginning of a sequence
- Circle with X - end of a "flow“ (not end of use case)
- “Target“ - entire use case is complete

Activity diagram - Notation

- ✓ Class
- ➢ Activity
- ● Use case

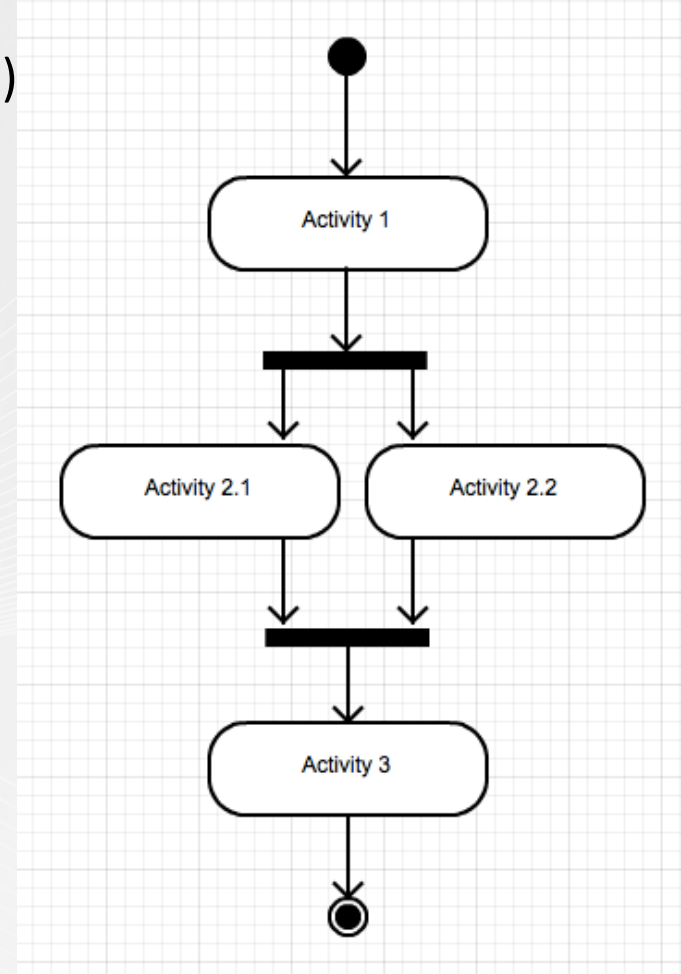
- Synchronization (Fork/Join)

- Fork

→ All activities can go in parallel

- Join

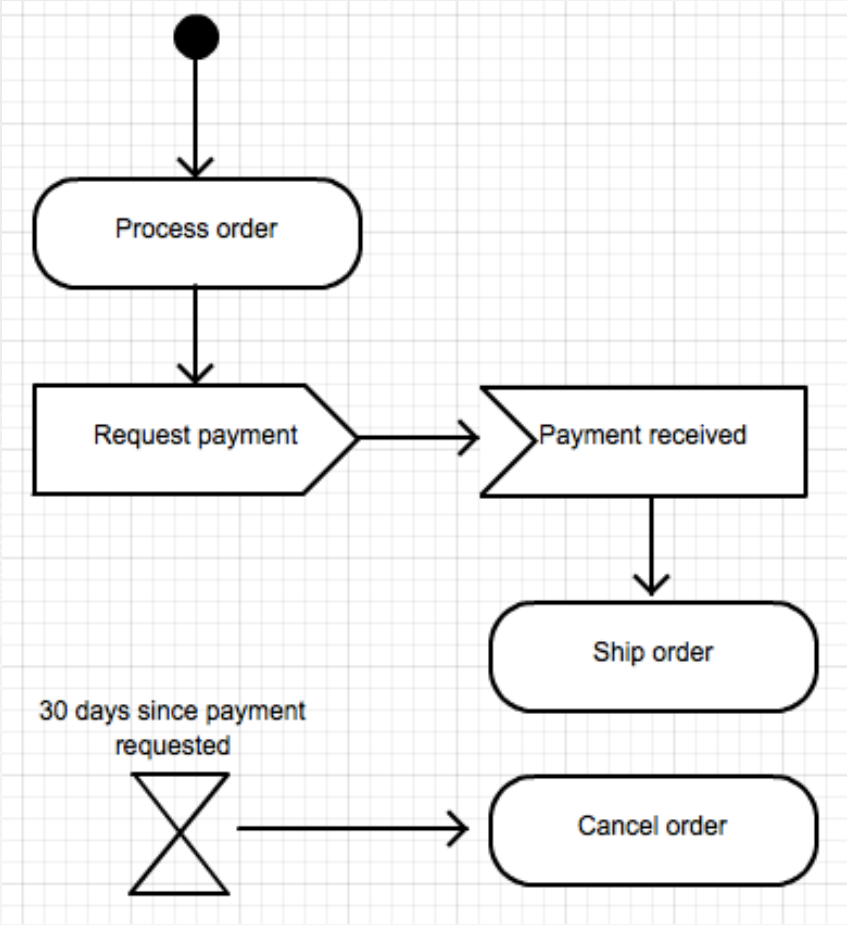
→ Progress cannot continue until all the activities that feed into the join complete



Activity diagram - Notation

- ✓ Class
- ➢ Activity
- ● Use case

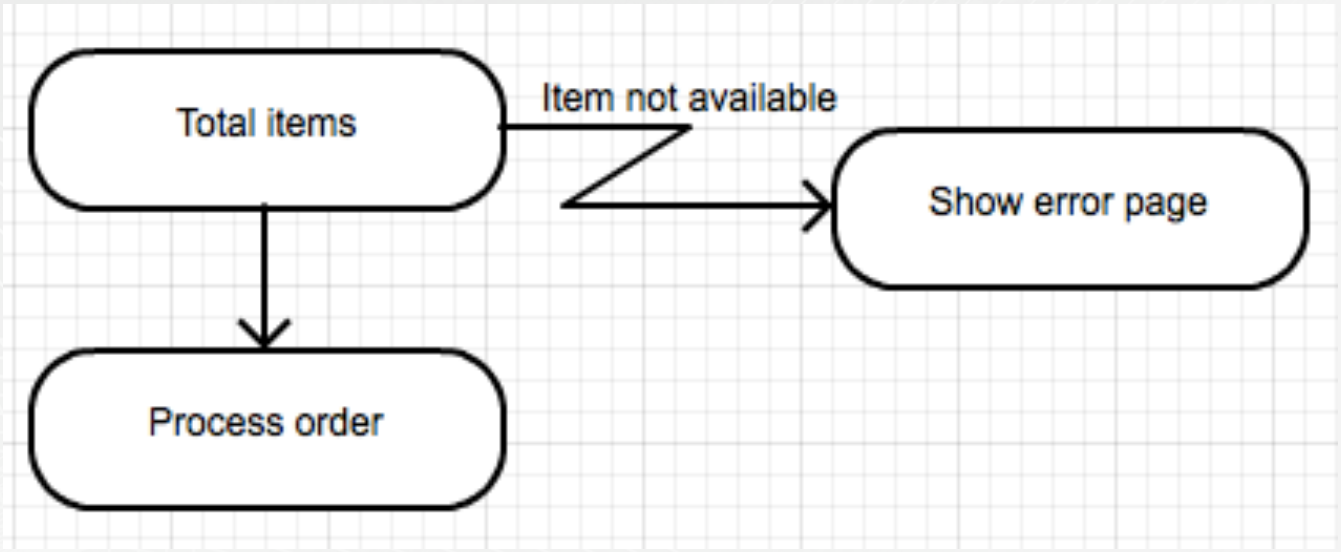
- Signals (Events)
- Generating signals
- Accepting signals
- Timer signals



Activity diagram - Notation

- ✓ Class
- ➢ Activity
- ● Use case

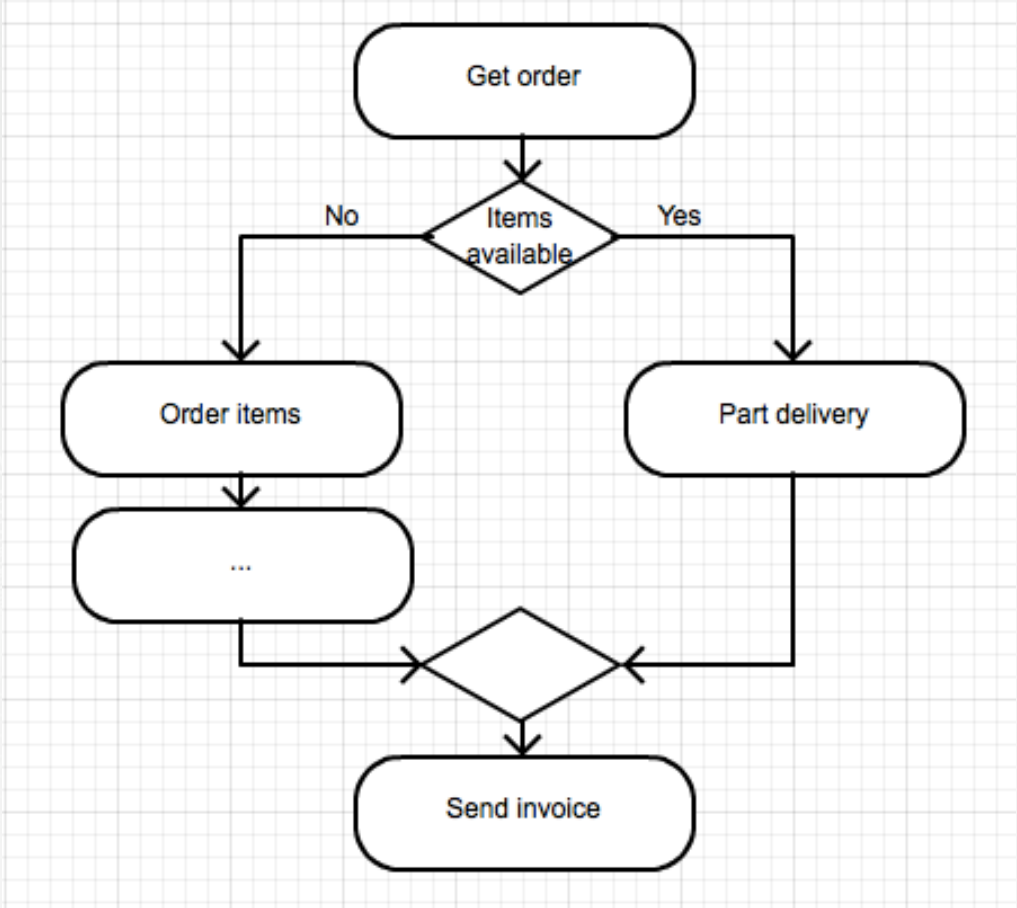
- Exception (Errors)



Activity diagram - Notation

- ✓ Class
- ➢ Activity
- ● Use case

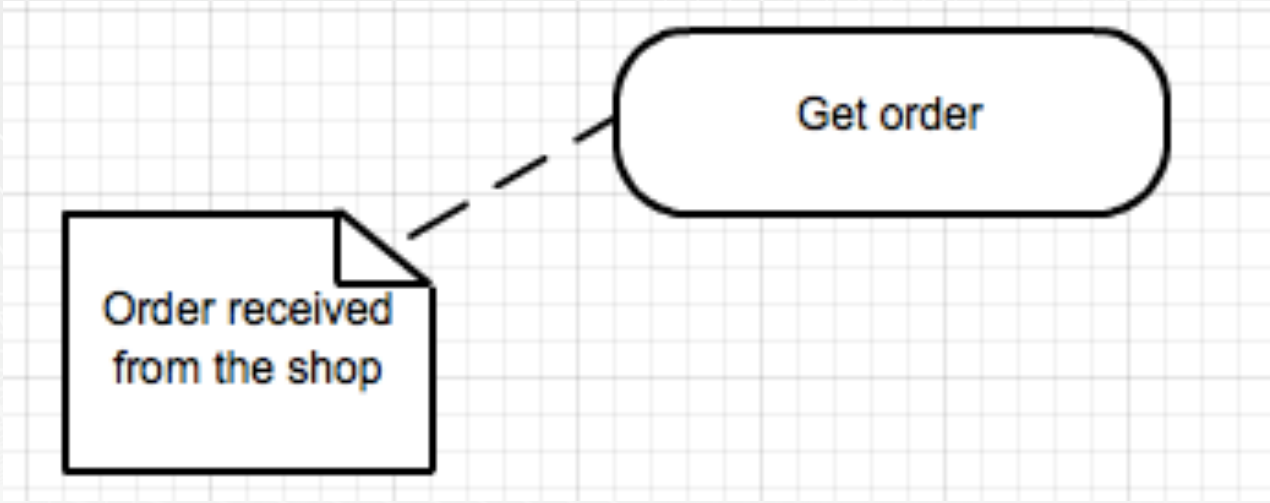
- Decision (Branch/Merge)



Activity diagram - Notation

- ✓ Class
- ➢ Activity
- ● Use case

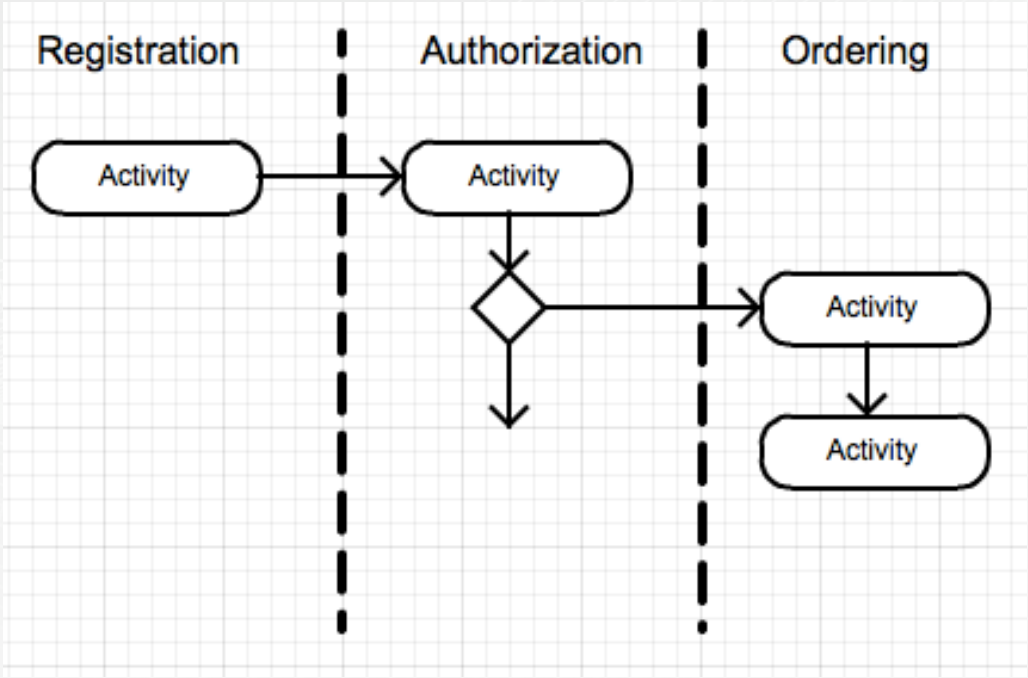
- Comments



Activity diagram - Notation

- ✓ Class
- ➢ Activity
- ● Use case

- Swin Lanes
- Each zone represents a area of responsibility
- Indicate who does what (Actors/Entities)



Activity diagram – Ad-/Disadvantages

- ✓ Class
- ➢ Activity
- ● Use case

- Advantages
- Easy to understand for employee and customer
- Describe parallel behavior
- Display multiple conditions and actors within a work flow by using swimlanes
- Disadvantages
- Use of swimlanes → a lot of actors/entities → diagram become very width
- Has the potential to become overly complex

  → New diagram for each work flow

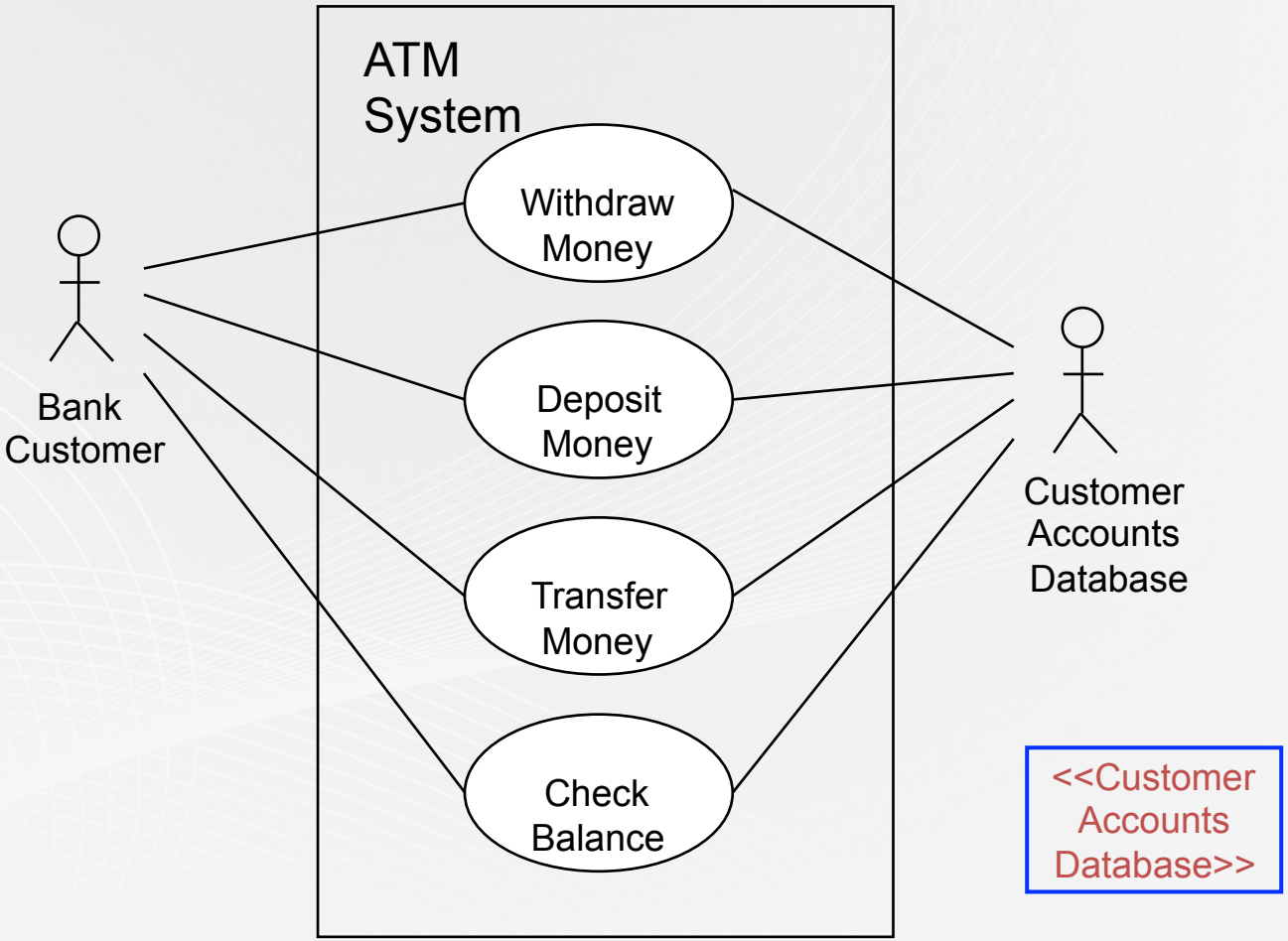
- ✓ Class
- ✓ Activity
- ➢ Use case

Use case diagram – General

- Visualizing relationships
  - Actors & Use cases
- Each ellipse in a UML use case diagram
  - → Functional requirement

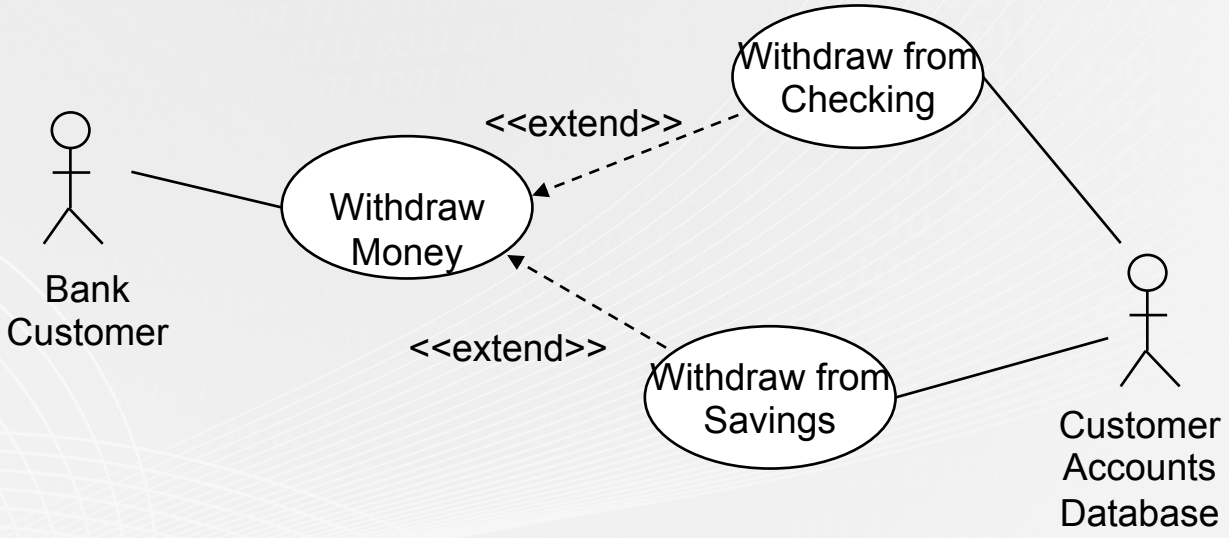
Use case diagram - Notation

- ✓ Class
- ✓ Activity
- ➢ Use case



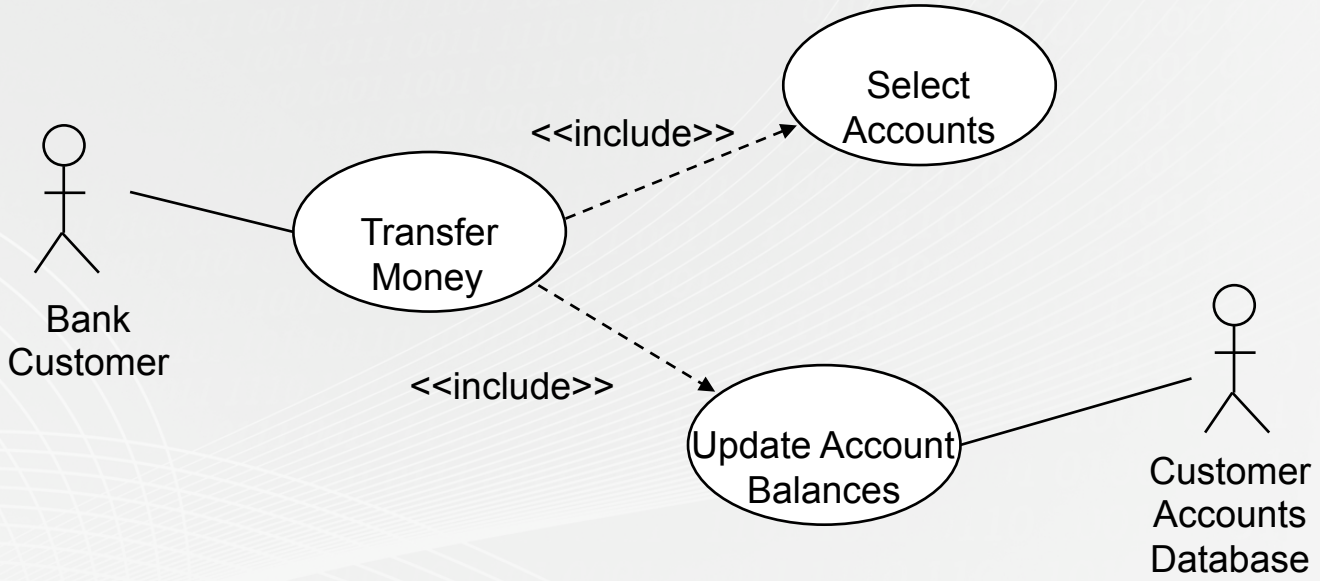
Use case diagram - Notation

- ✓ Class
- ✓ Activity
- ➢ Use case



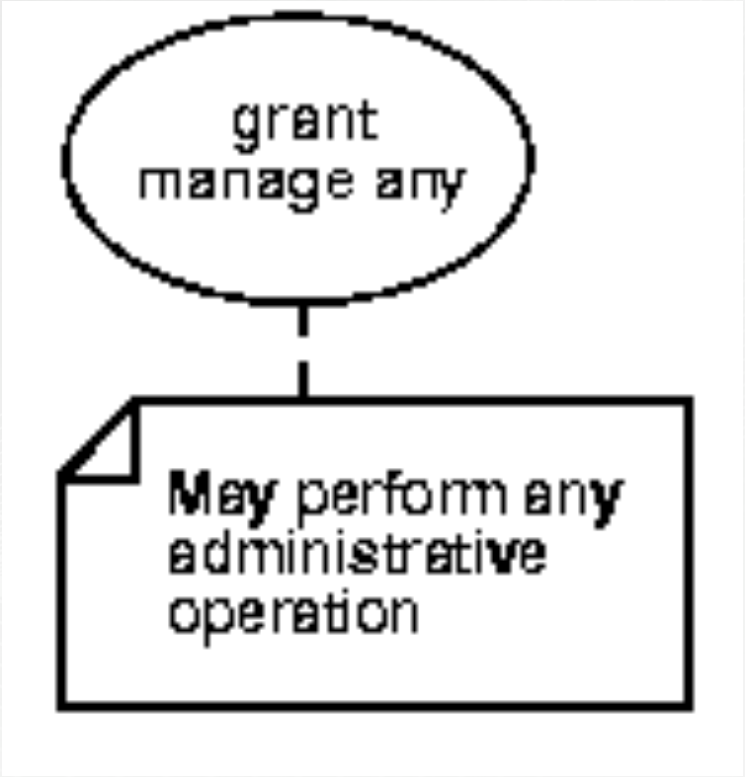
Use case diagram - Notation

- ✓ Class
- ✓ Activity
- ➢ Use case



Use case diagram - Notation

- ✓ Class
- ✓ Activity
- ➢ Use case



- ✓ Class
- ✓ Activity
- ➢ Use case

Use case diagram – Ad-/Disadvantages

- Advantages
- Summary of the whole software system
- Feedback at a very early stage
- Other aspects of software development

- Disadvantages
- Not Capture the non-functional requirements easily

Any questions?

Thanks for your attention!