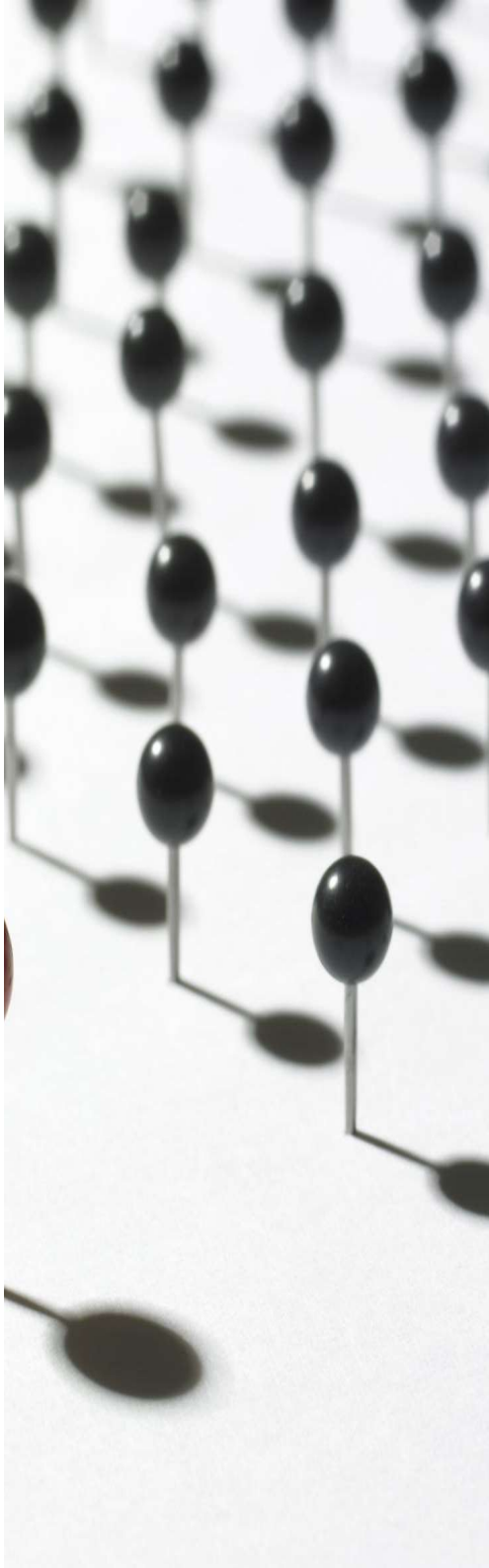
# UML and software analys

# Objectives

- Introduction to UML
- In the end, you should
  - Be able to identify diagram types
  - Interpret a diagram
  - Design correct diagrams
- You will not learn how to design
  - Nor Model Driven Engineering

# Why model ?

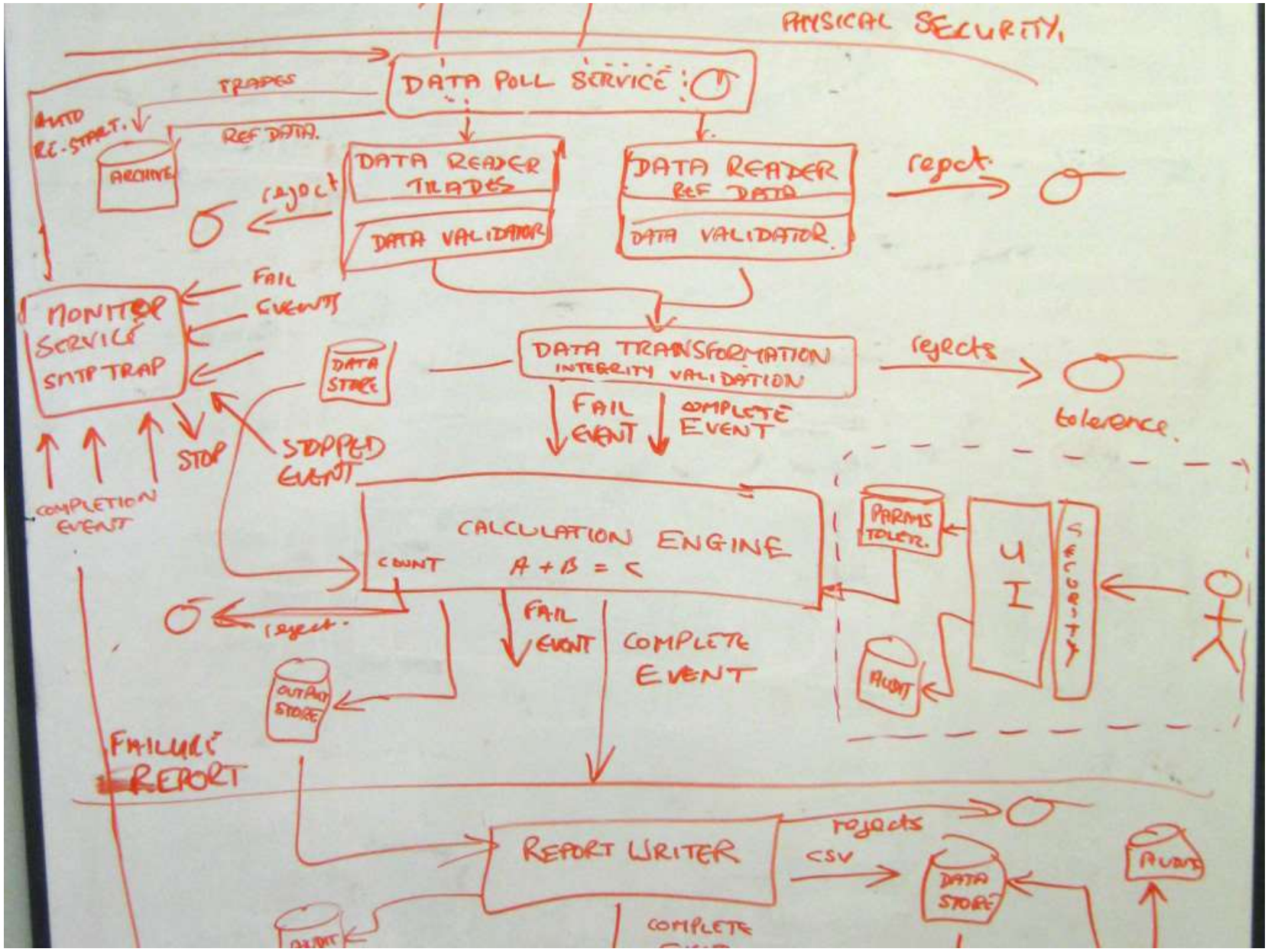
- Convey abstraction
- Communicate without unnecessary details
- Help to deal with complexity



# Who model ?

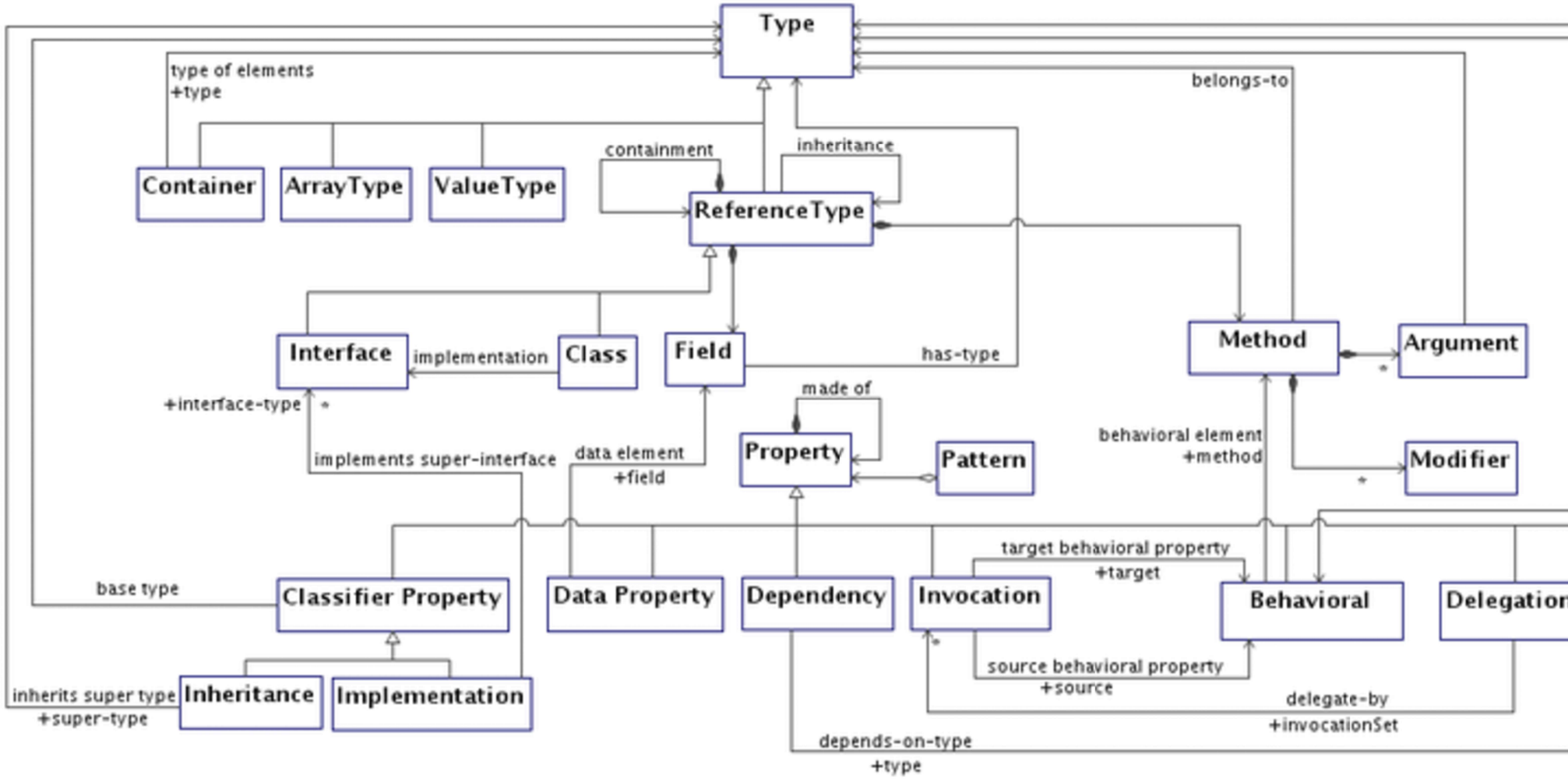
- Virtually no one

# A Software architecture Model




https://c4

# We need models, we like models, we love metamodels

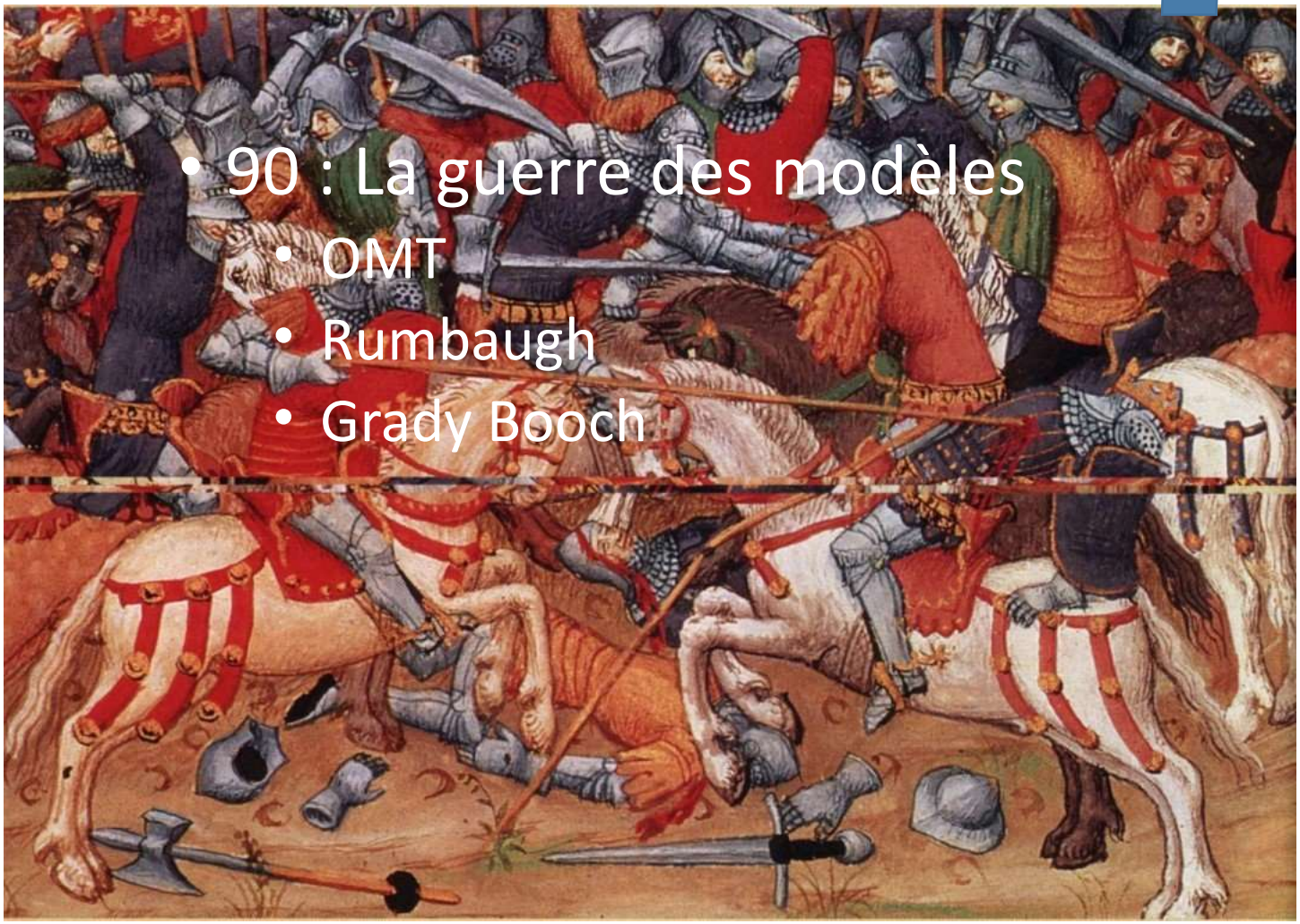


# Before UM

- In the 90's, apparit
  modeling paradigr
  - OMT (James R
    oriented towa
  - Booch (Grady
    design oriente
  - OOSE (Ivar Jac

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

# War of models

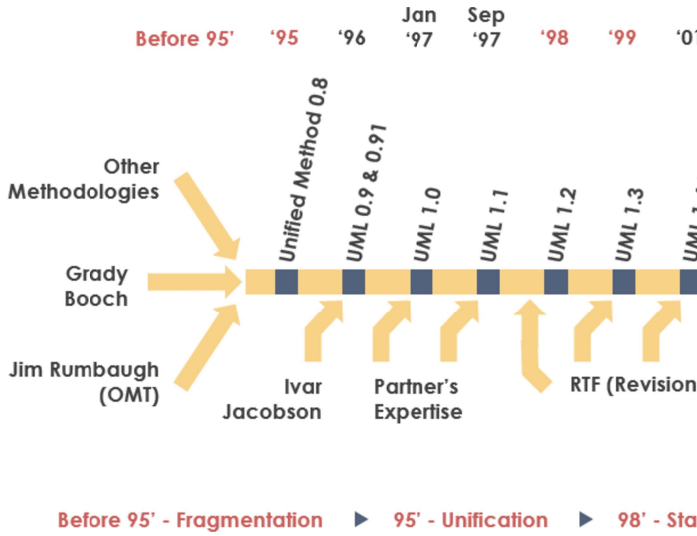




1995

# Unified Modeling Language

- Need to standardize modeling languages
- 1994 = End of the models war
  - Unification of Rumbaugh and Booch propositions
  - Jacobson join the band (The three amigos)
- Quick success
- OMG (Object Management Group) take charge of the standard
  - www.omg.org
- Version 2.5 in 2013



# Definition

- UML is a textual and graphical modeling language to
  - Understand and design requirements,
  - communicate
  - specify,
  - document,
  - architect,
  - design.
- Used mostly for software intensive systems
  - bank, telecommunications, industry, avionics, healthcare, …

# 13 diagrams

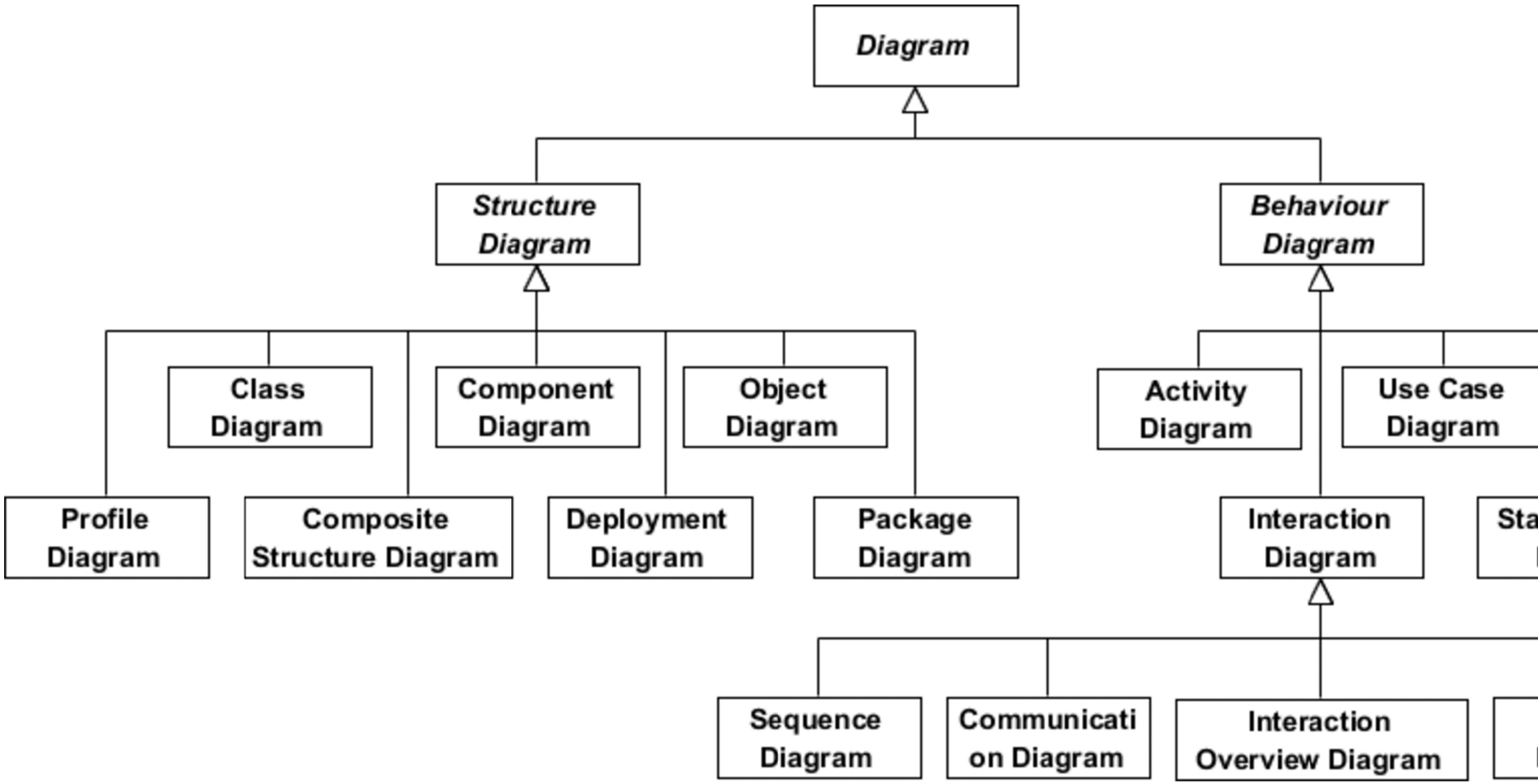
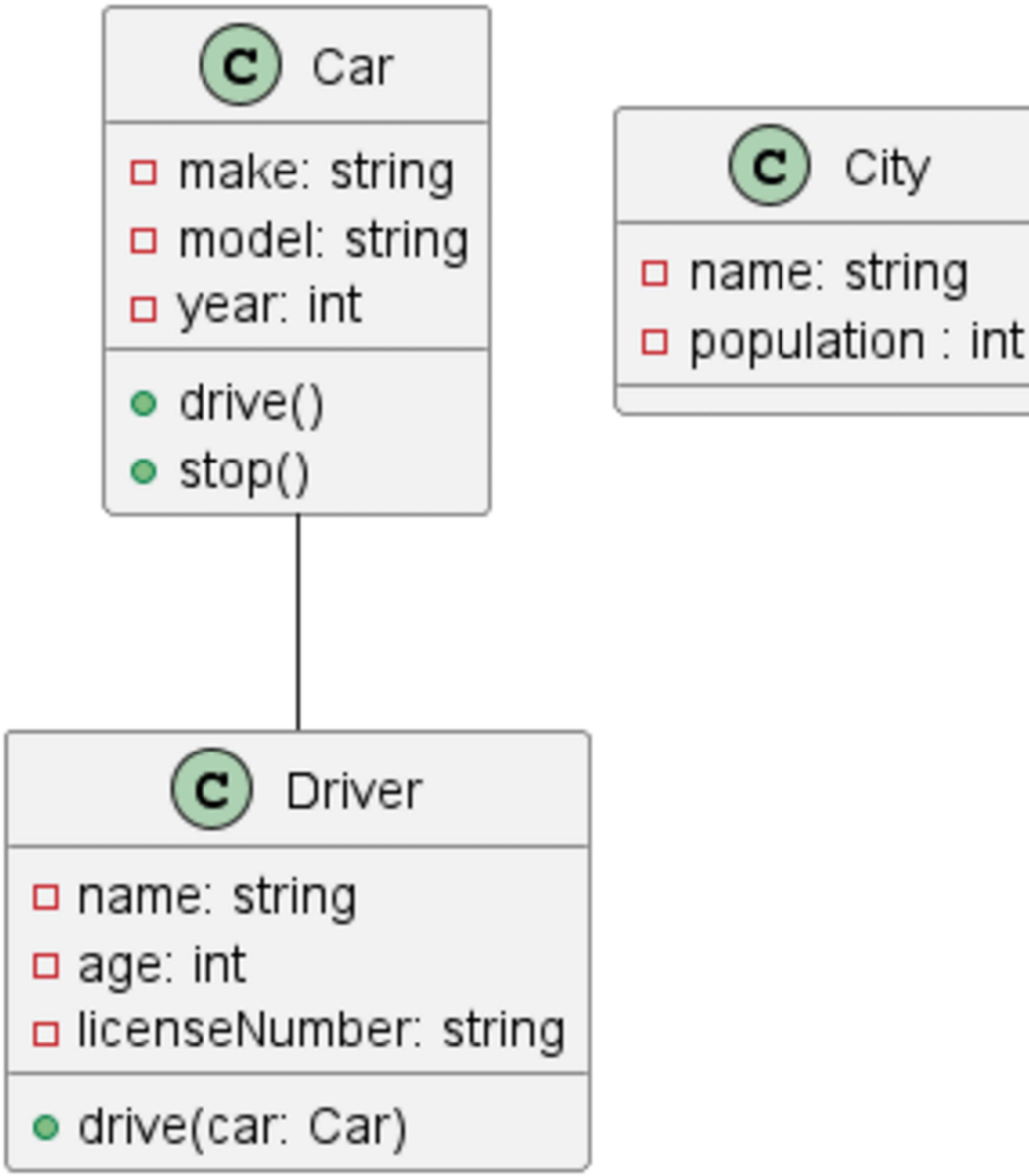


Diagram
Structure Diagram
Behaviour Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
Profile Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
Sta
Sequence Diagram
Communicati on Diagram
Interaction Overview Diagram

# Class Diagram

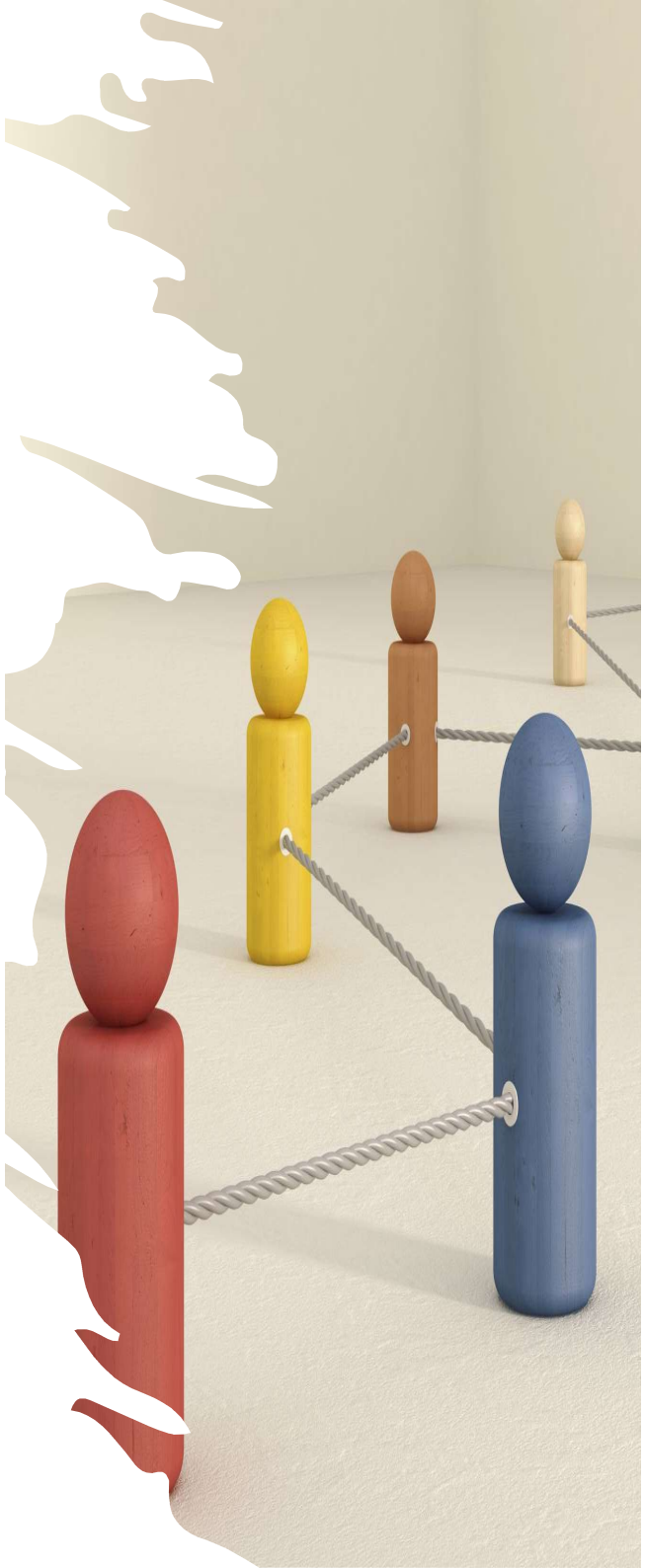
- Structure diagram
- Modeling of classes and entities
  - Attributes
  - Operations
  - Relations
  - Inheritance
  - Types

# Class notations



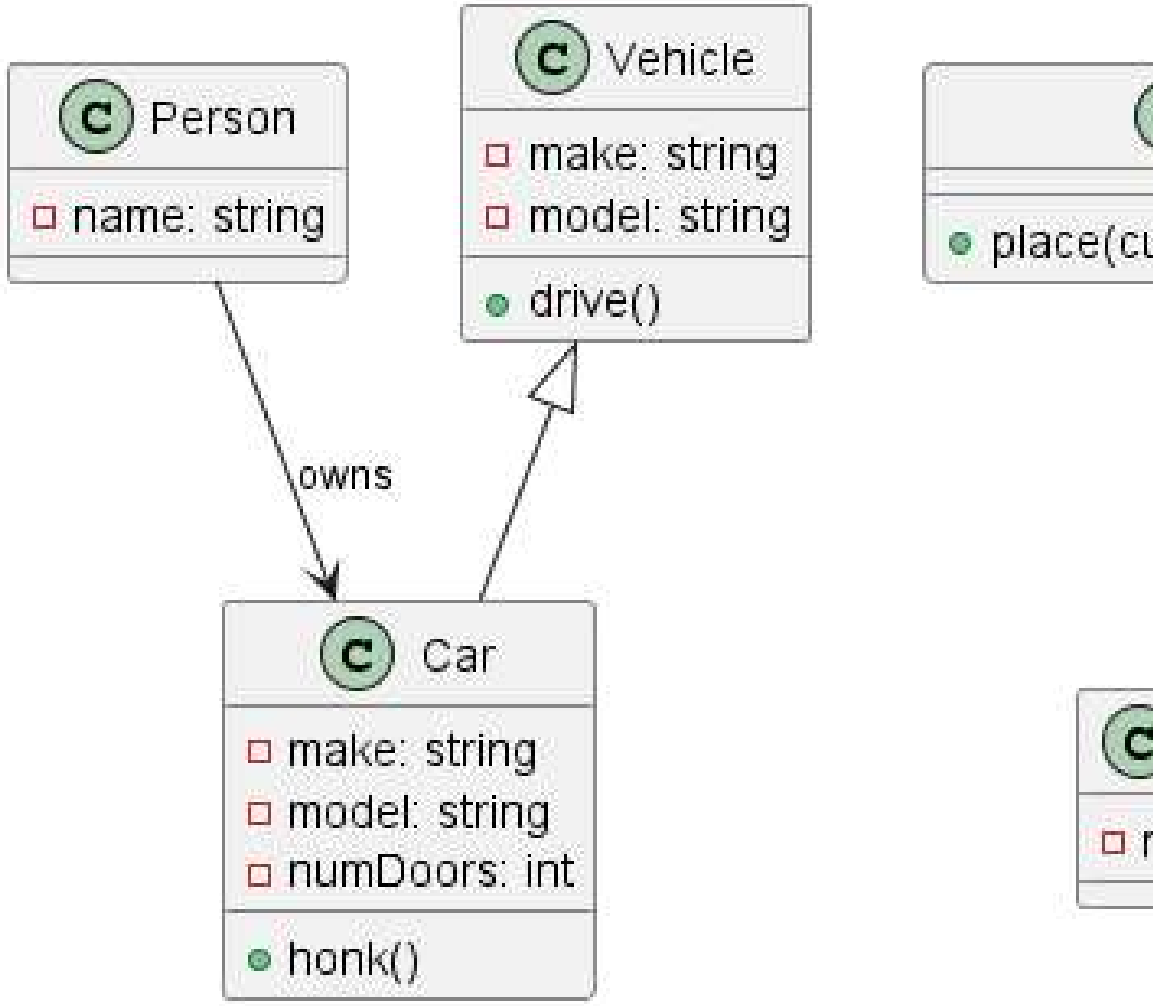
# Relations

- In UML, relationships describe how different elements of a system are connected to each other. There are several types of relationships, each with a specific meaning and notation.

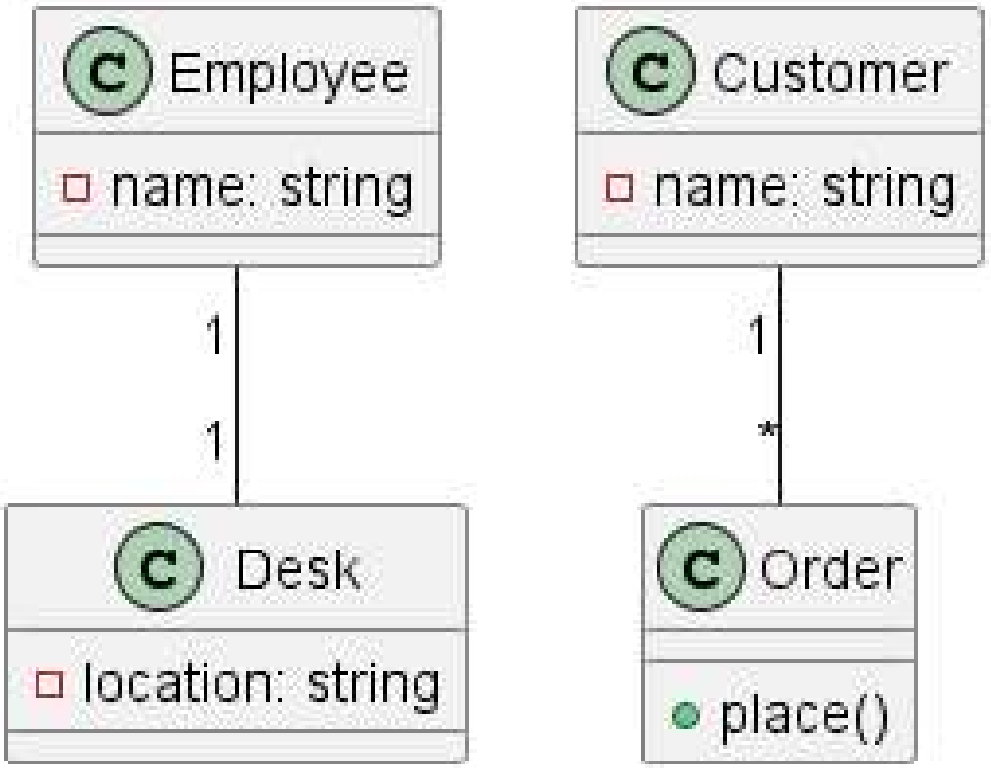


# Different type of relationships

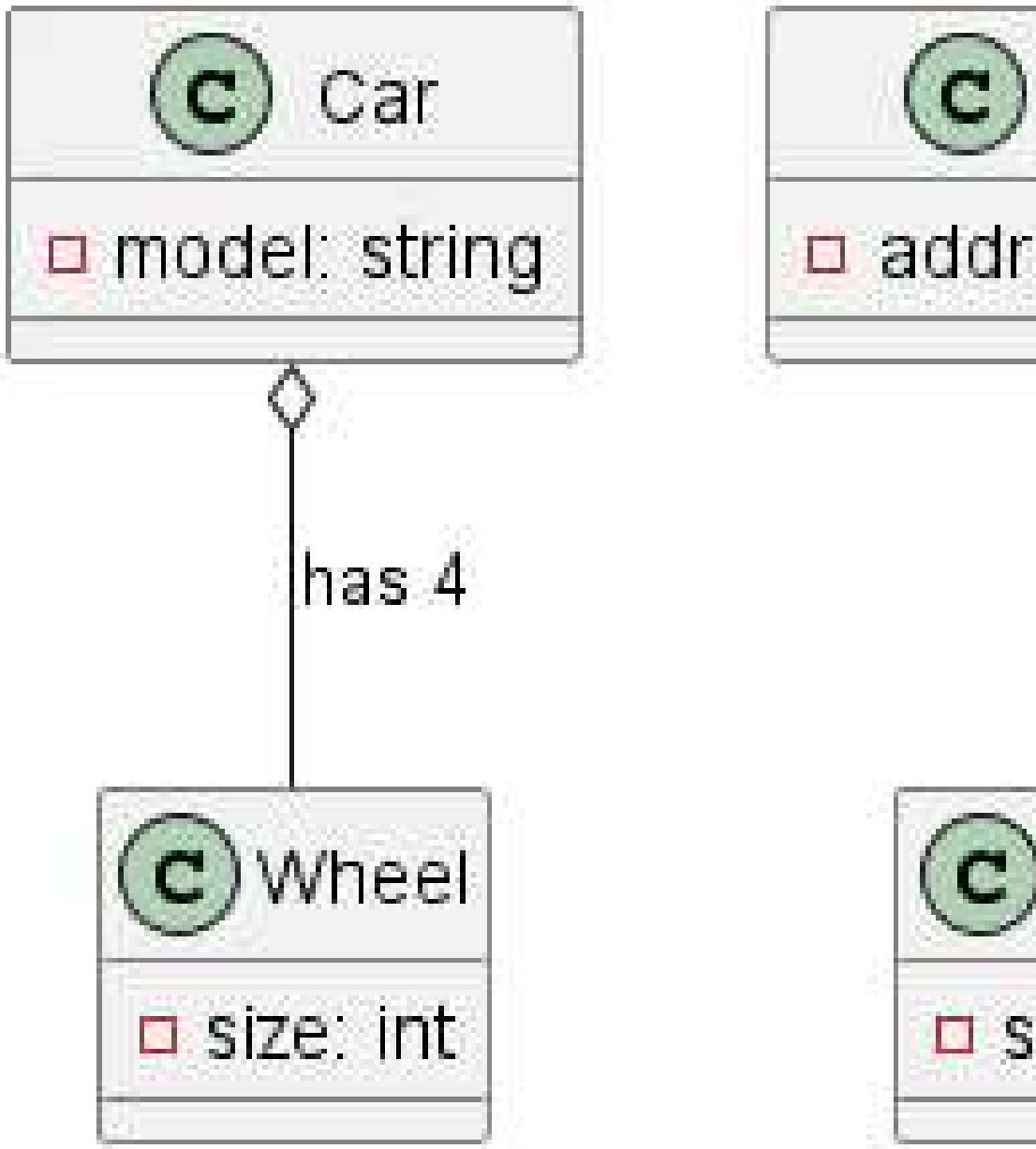
- Association Relationship
- Generalization Relationship
- Dependency Relationship



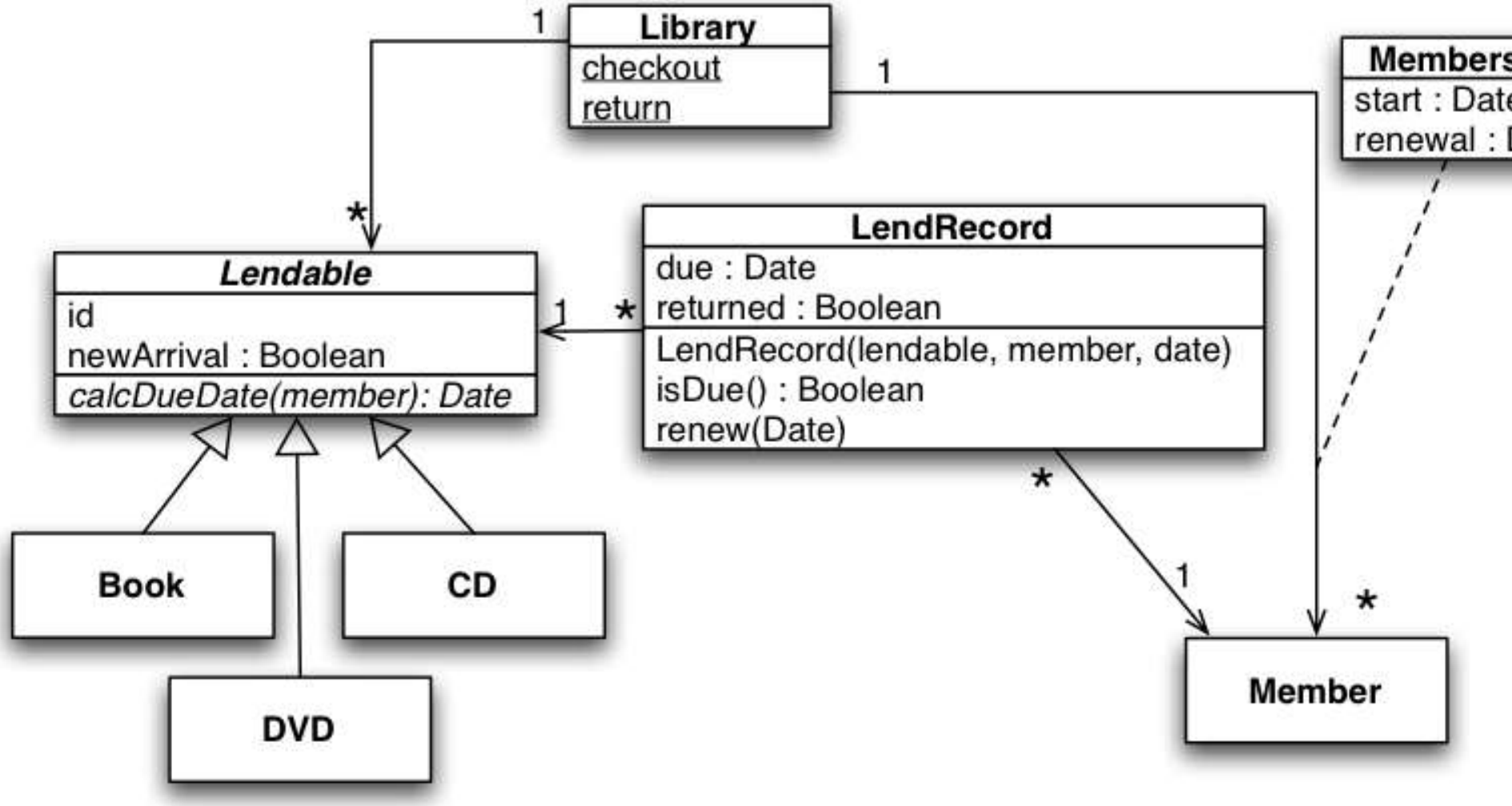
# Cardinalities



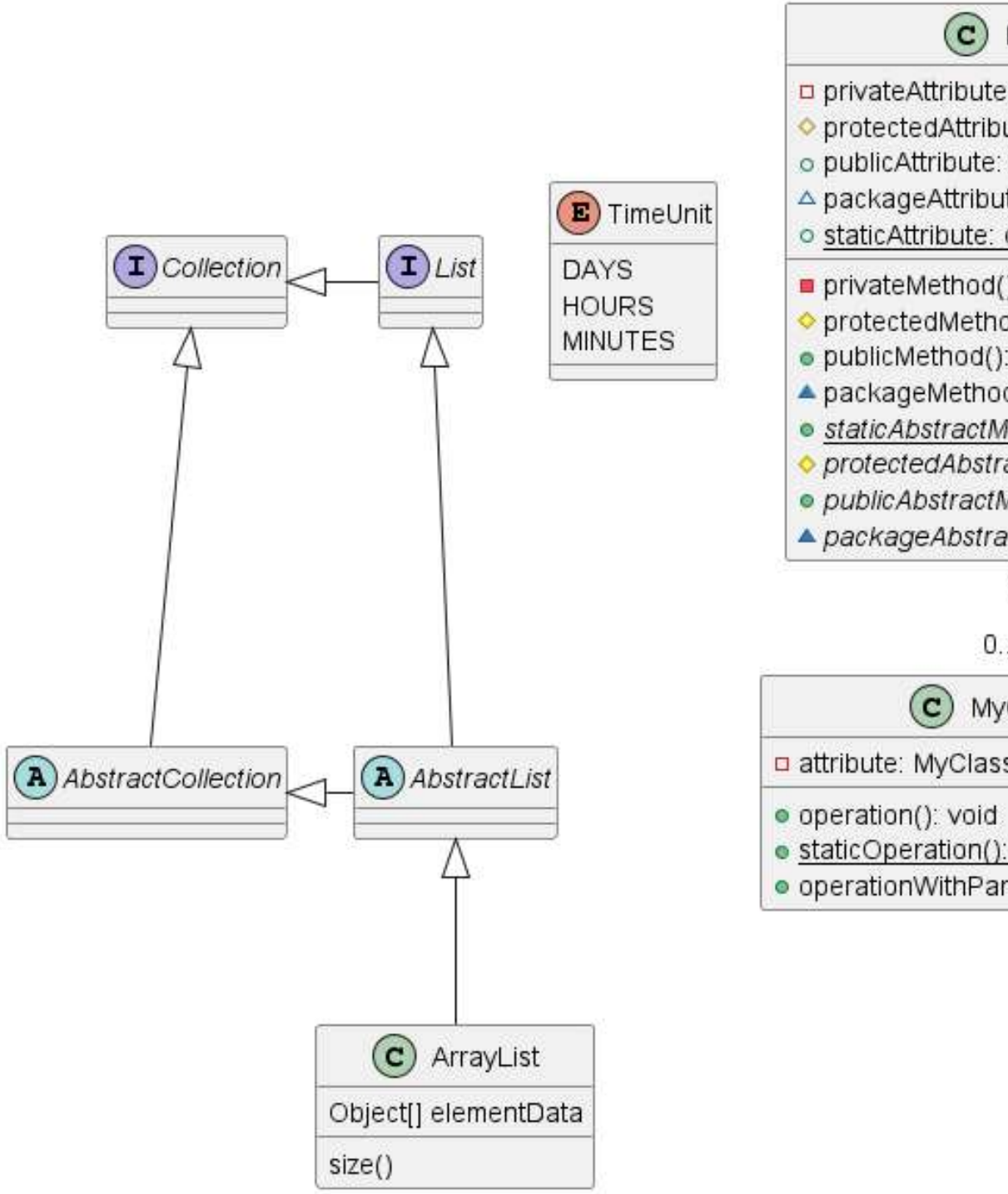
# Aggregation vs Composition



# Example: Library Classes

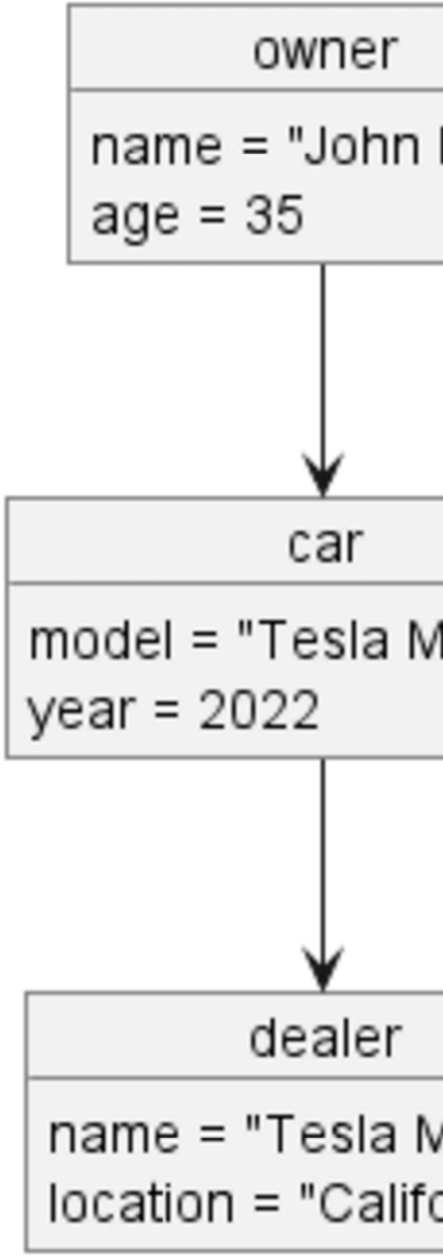


# Notations


Collection
List
TimeUnit
DAYS
HOURS
MINUTES
AbstractCollection
AbstractList
ArrayList
Object[] elementData
size()
privateAttribute
protectedAttribu
publicAttribute:
packageAttribut
staticAttribute:
privateMethod(
protectedMetho
publicMethod():
packageMethod
staticAbstractM
protectedAbstra
publicAbstractM
packageAbstra
0..
My
attribute: MyClass
operation(): void
staticOperation():
operationWithPar

# Object Diagram

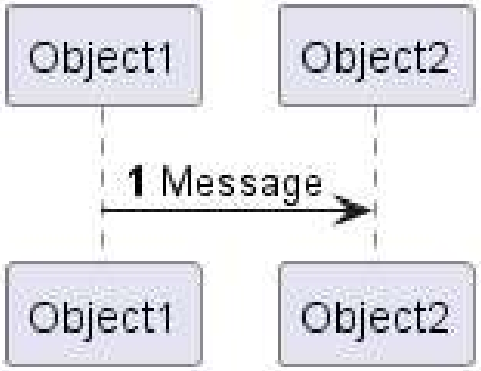
- An object diagram is a type of structural diagram in UML that shows a snapshot of the instances of classes in a system at a particular moment in time.
- It represents the state of objects in the system, along with their relationships and attributes.
- An object diagram can be seen as an instance-level view of a class diagram.
- It is helpful in understanding complex systems, as it provides a visual representation of the current state of the system

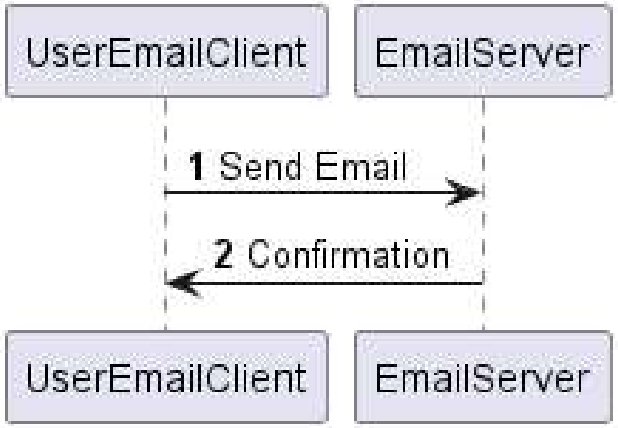


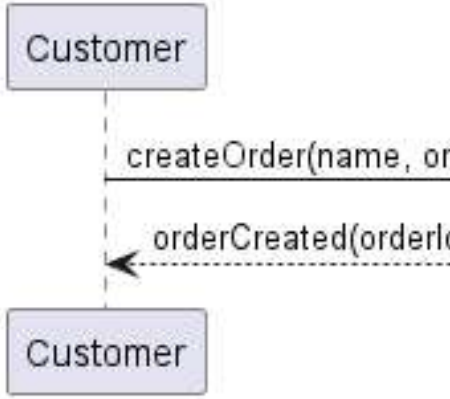
# Sequence Diagram

- This diagram emphasises chronology.
  - Modeling the dynamic aspects of re
    systems and scenarios involving a sr
    objects.
  - A rectangle and a lifeline represent
  - Messages are represented by arrow
    the sender to the receiver.
- They are used in two ways, depending o
  development phase.
  - Documentation of use cases
  - Precise representation of interactio
    objects and control flow distributio

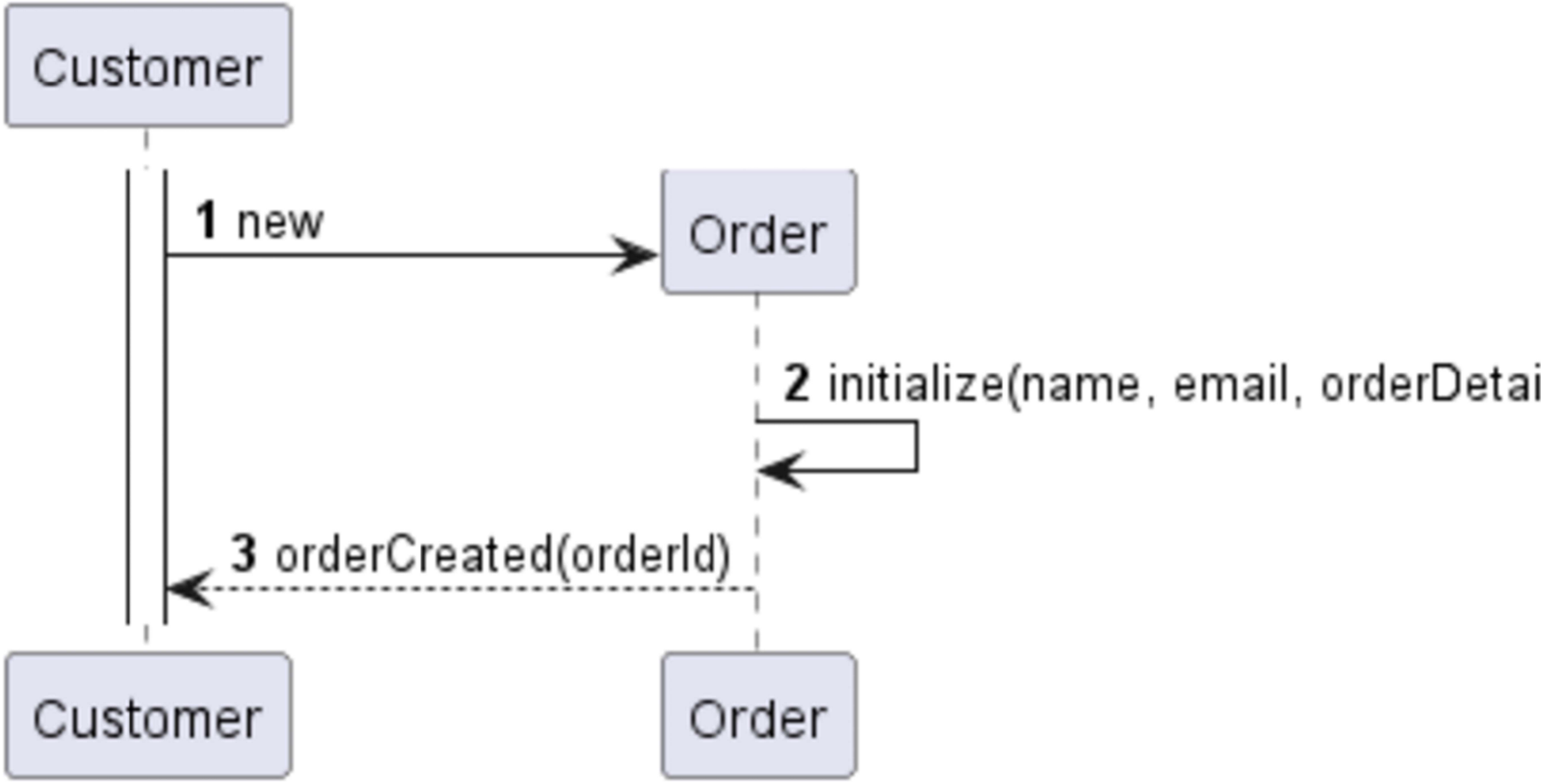
# Sending a message



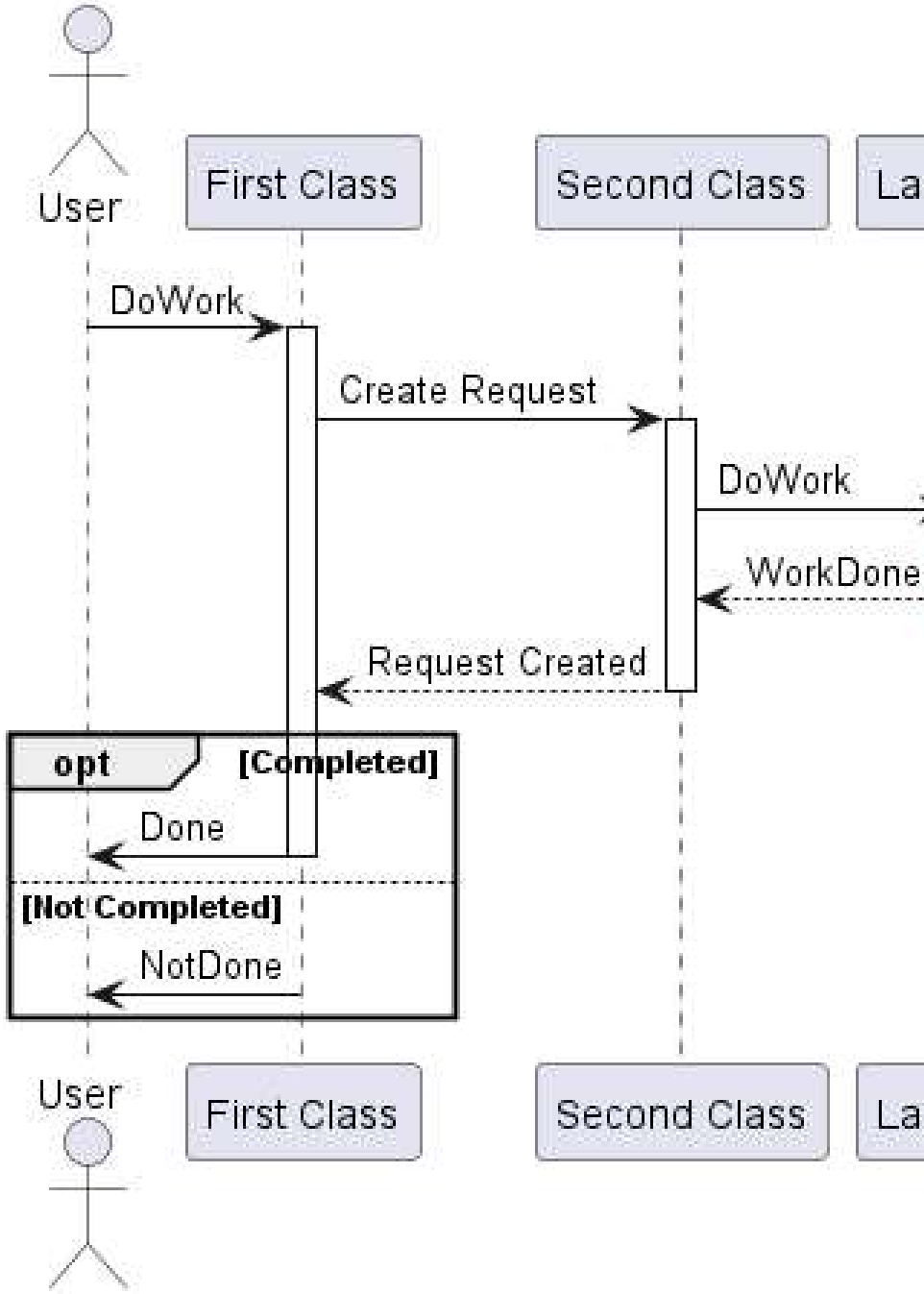



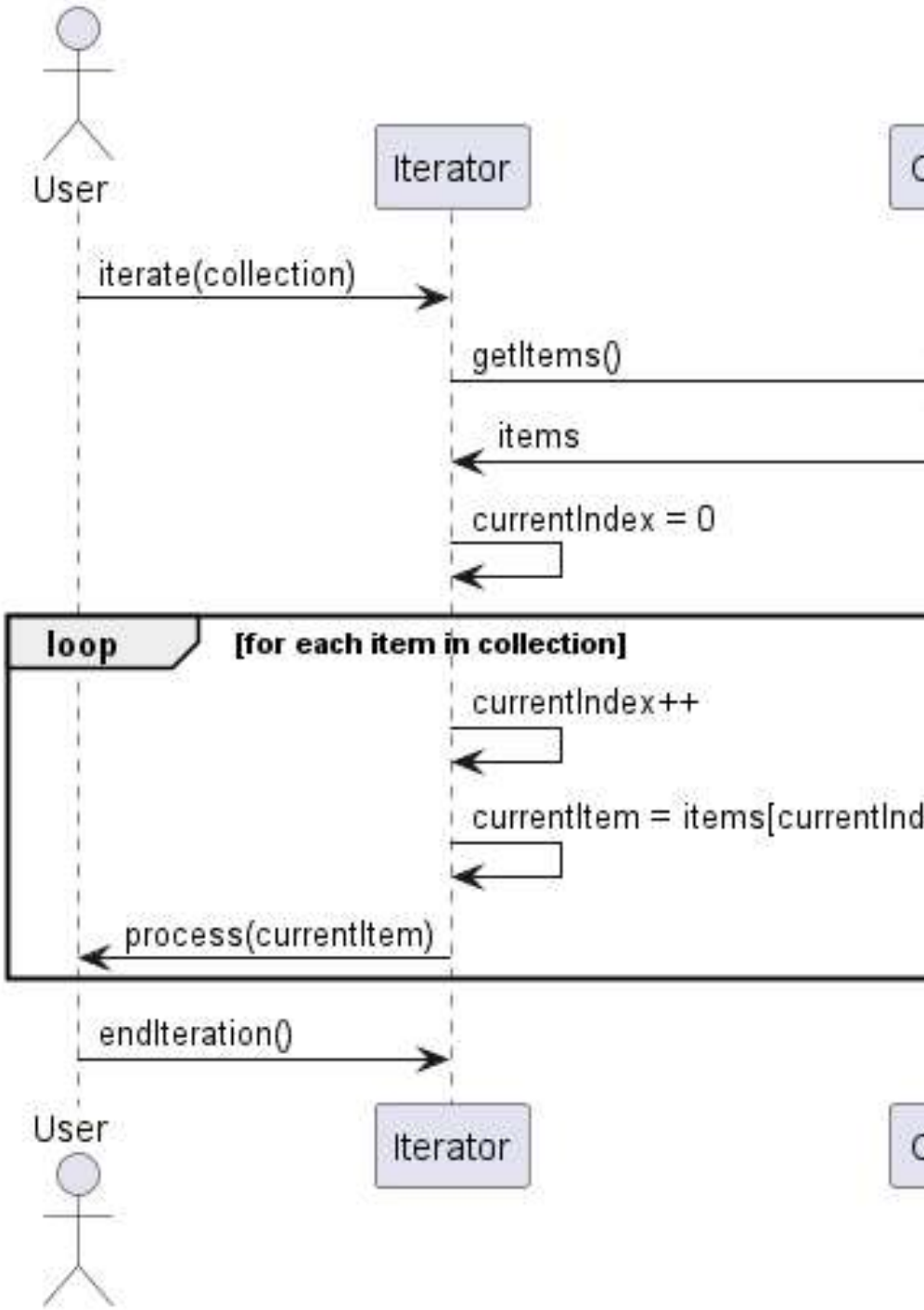
# Sending a message to create an object



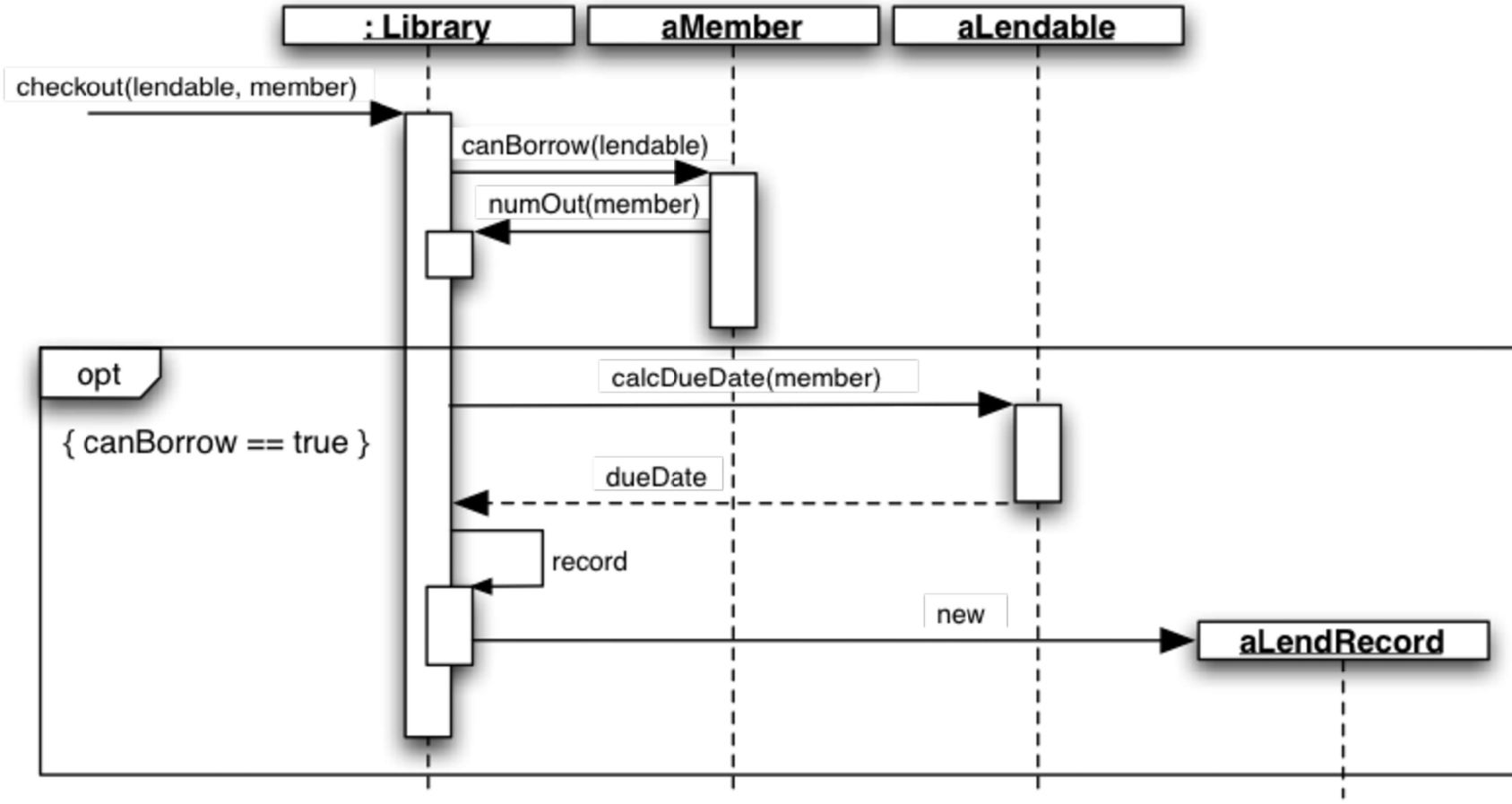
# Example with an option bloc



# Example with an Iteration

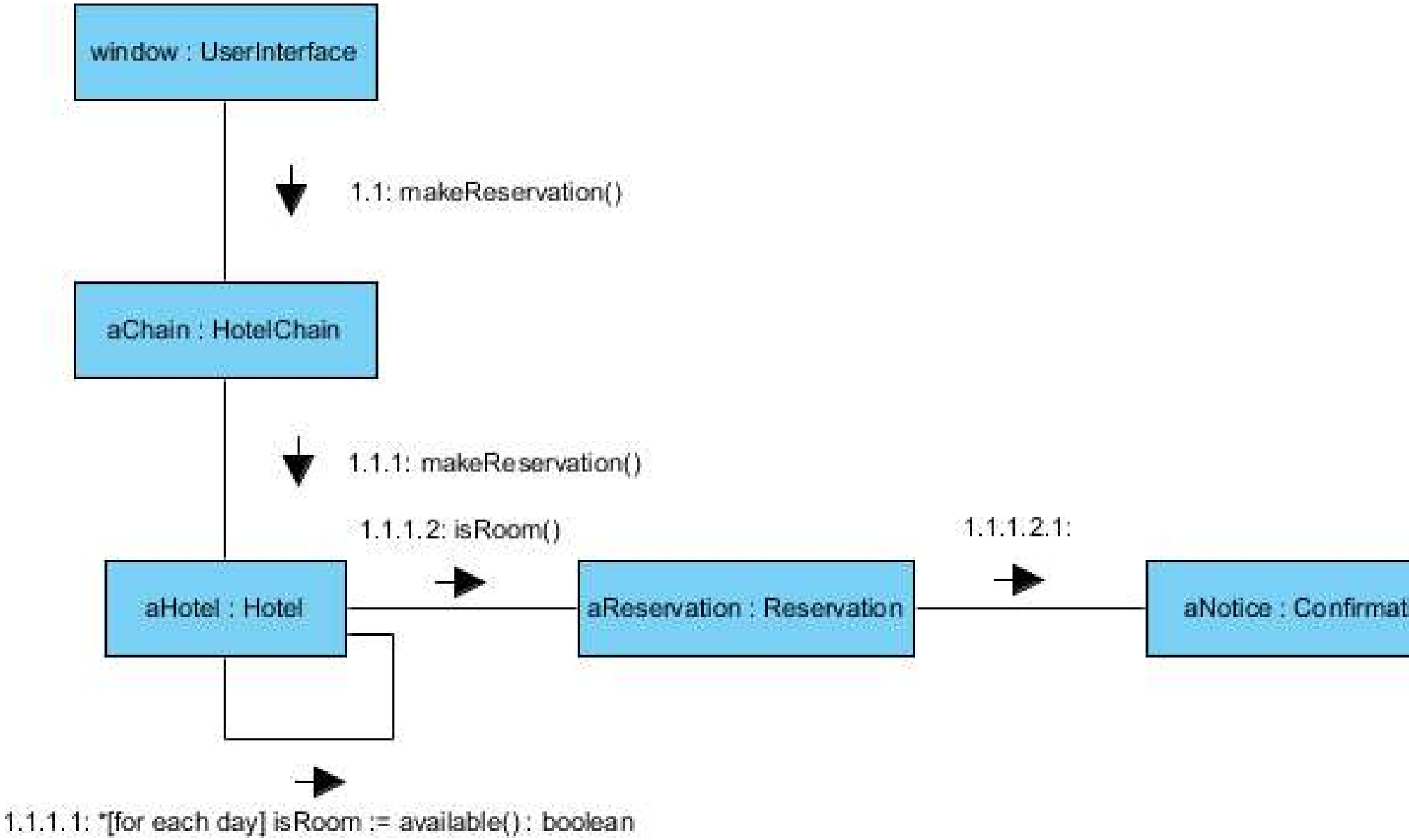


# The Library

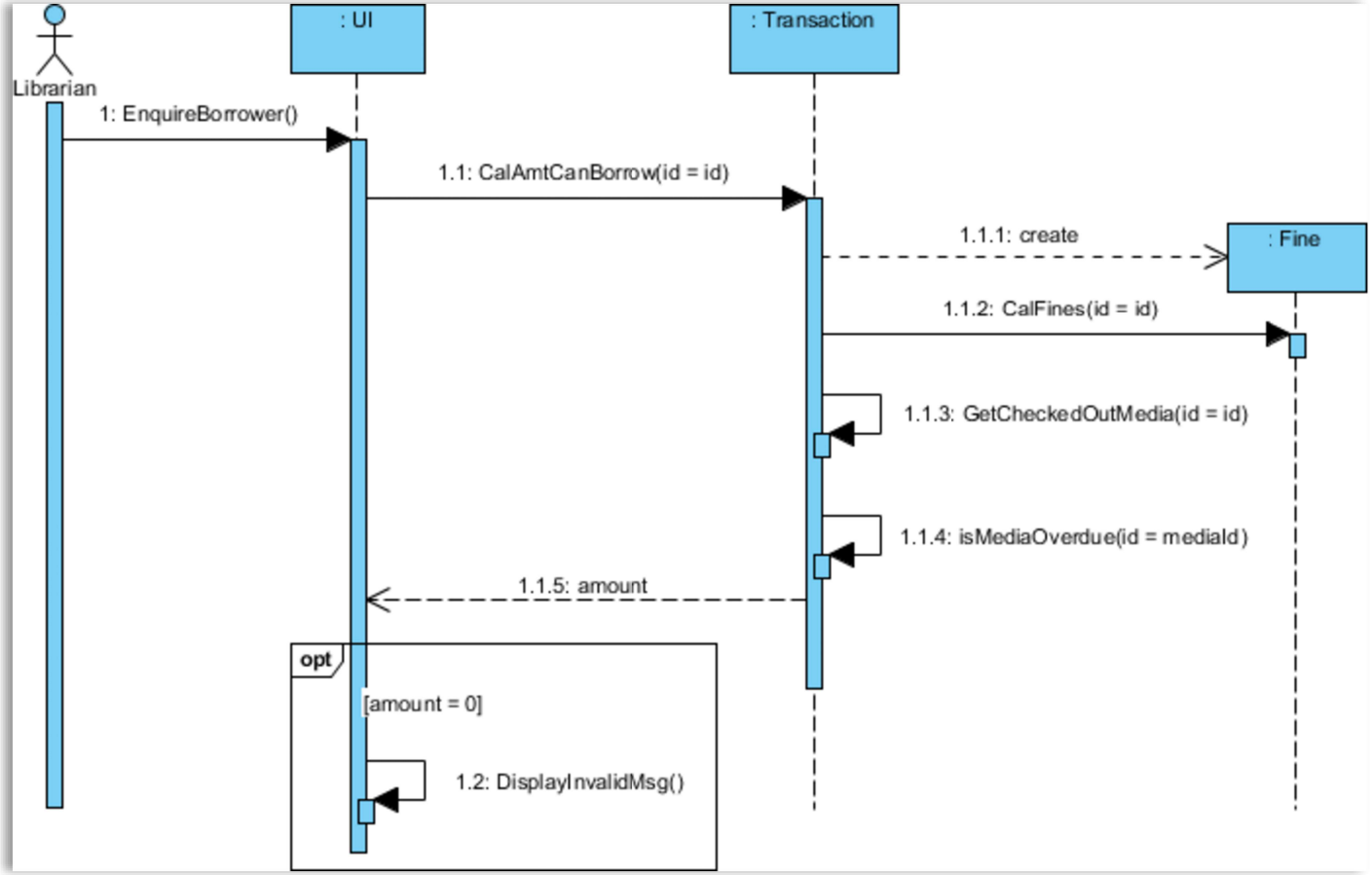

: Library
aMember
aLendable
checkout(lendable, member)
canBorrow(lendable)
numOut(member)
opt
{ canBorrow == true }
calcDueDate(member)
dueDate
record
new
aLendRecord

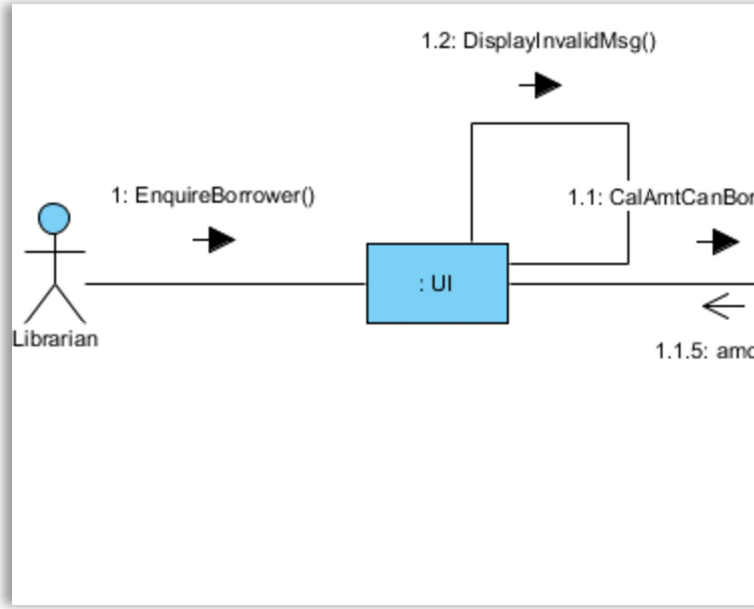
# Communication Diagram

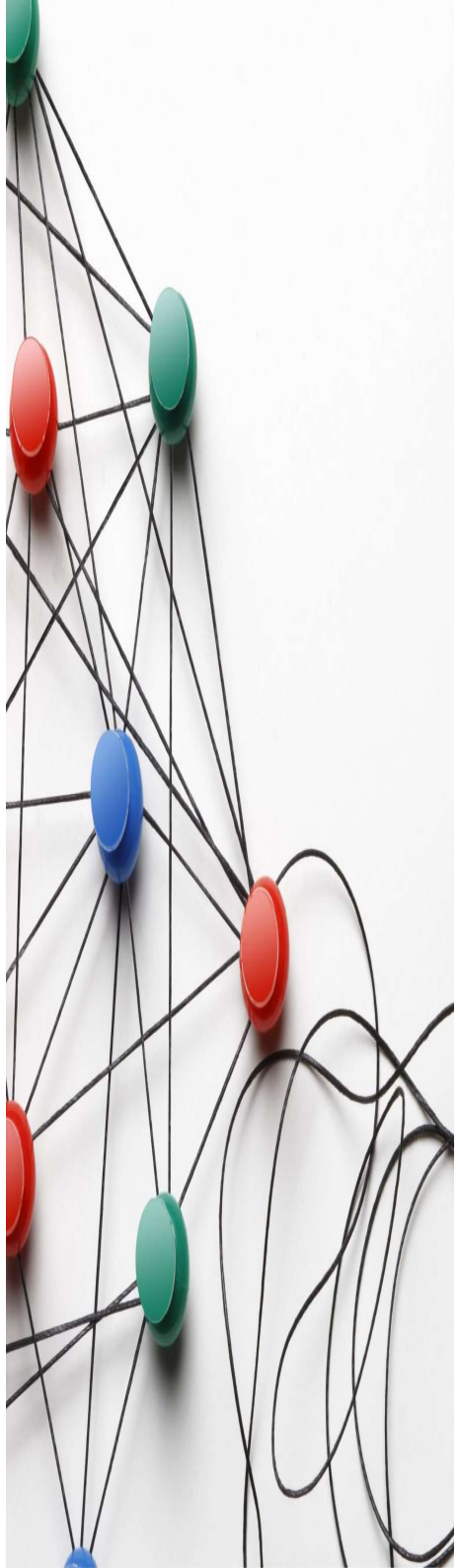
- Used to model the interactions between objects or components in a system.
- They are also known as "collaboration diagrams" or "interaction diagram
- Show the relationships between objects or components in a system, as as the messages that are exchanged between them.
- The objects or components are represented as rectangles,
- The messages exchanged between objects or components are represen as arrows, with the name of the message and any arguments or parameters included on the arrow.
- Powerful tool for visualising the interactions between objects or components in a system, and can help developers identify and resolve system behaviour and performance issues.


window : UserInterface
1.1: makeReservation()
aChain : HotelChain
1.1.1: makeReservation()
1.1.1.2: isRoom()
1.1.1.2.1:
aHotel : Hotel
aReservation : Reservation
aNotice : Confirmat
1.1.1.1: *[for each day] isRoom := available() : boolean

# Sequence vs communication





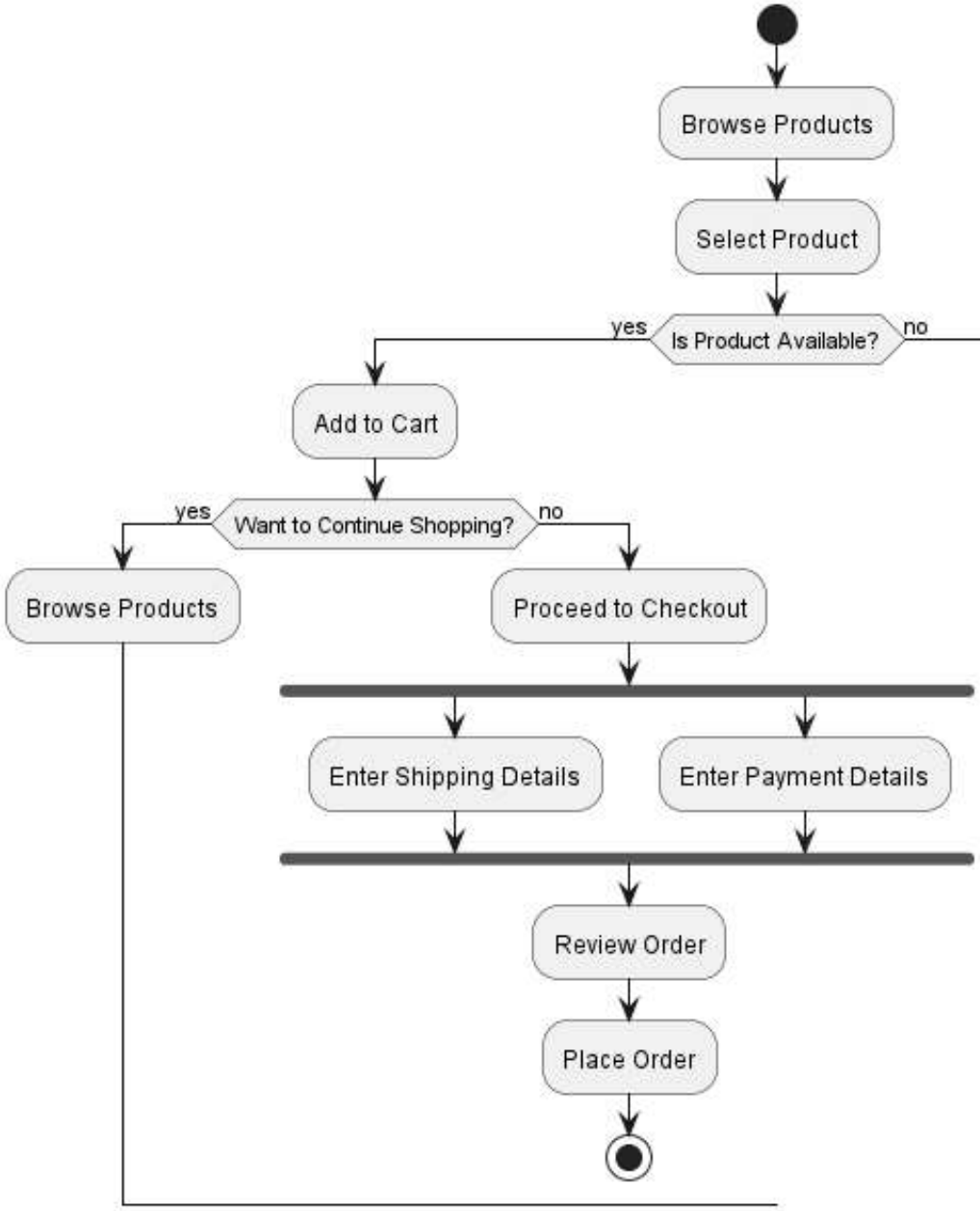
# Activity Diagra

- A graphical representa workflows
- Models the dynamic be system
- Captures the flow of co between activities
- Can represent concurre parallel behavior
- Visualizes the sequenc in a process

# Key Elements

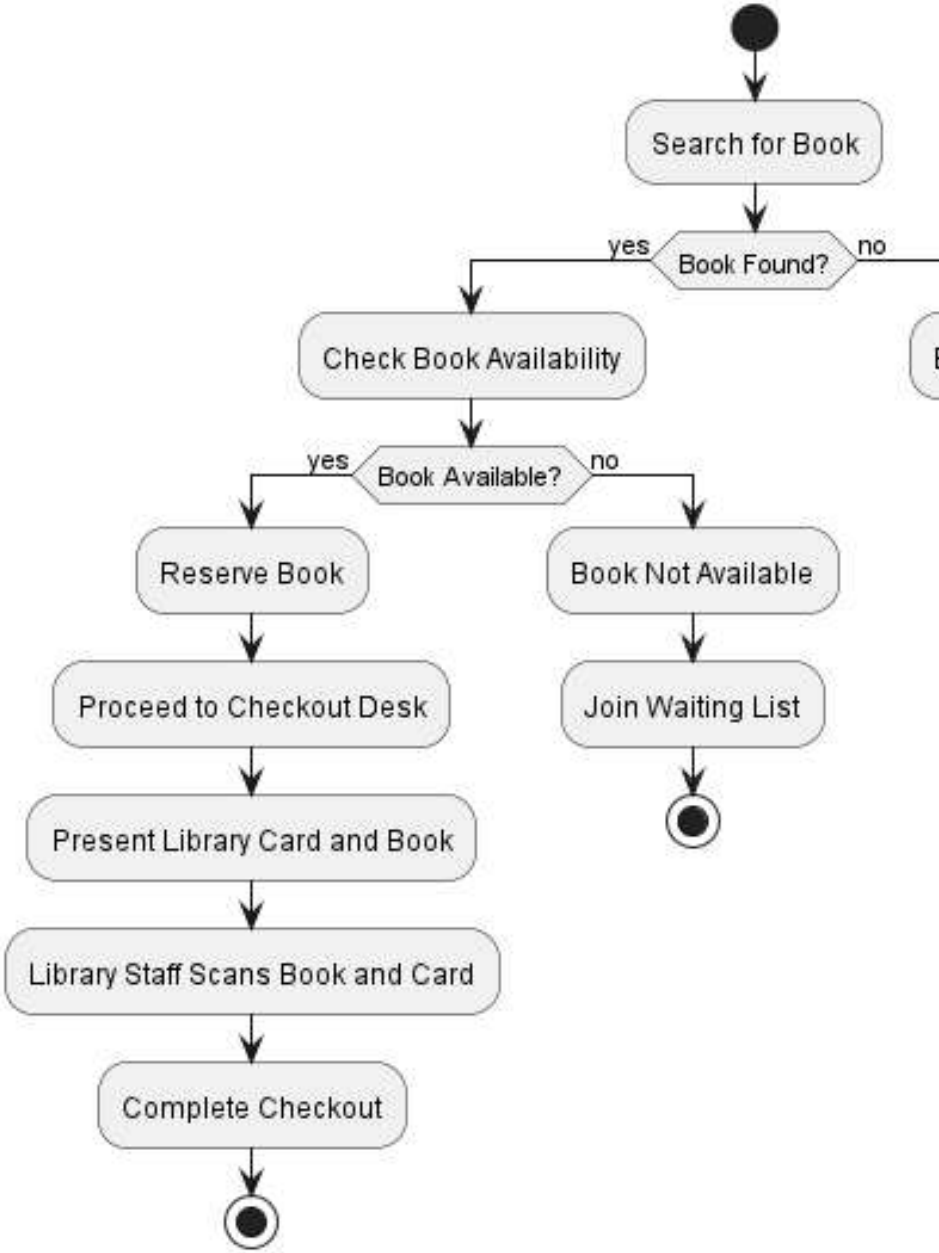
- Activity: A unit of work or operation performed by the system
- Control Flow: Arrows representing the flow of control between activities
- Decision Node: A diamond shape representing a decision point
- Merge Node: Merging multiple control flows into a single flow
- Fork Node: Splitting a single control flow into multiple parallel flows
- Join Node: Synchronizing multiple parallel flows into a single flow
- Initial Node: The starting point of the activity diagram
- Final Node: The end point of the activity diagram

# Online shopping activity diagram




https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-

# Library checkout activity diagram



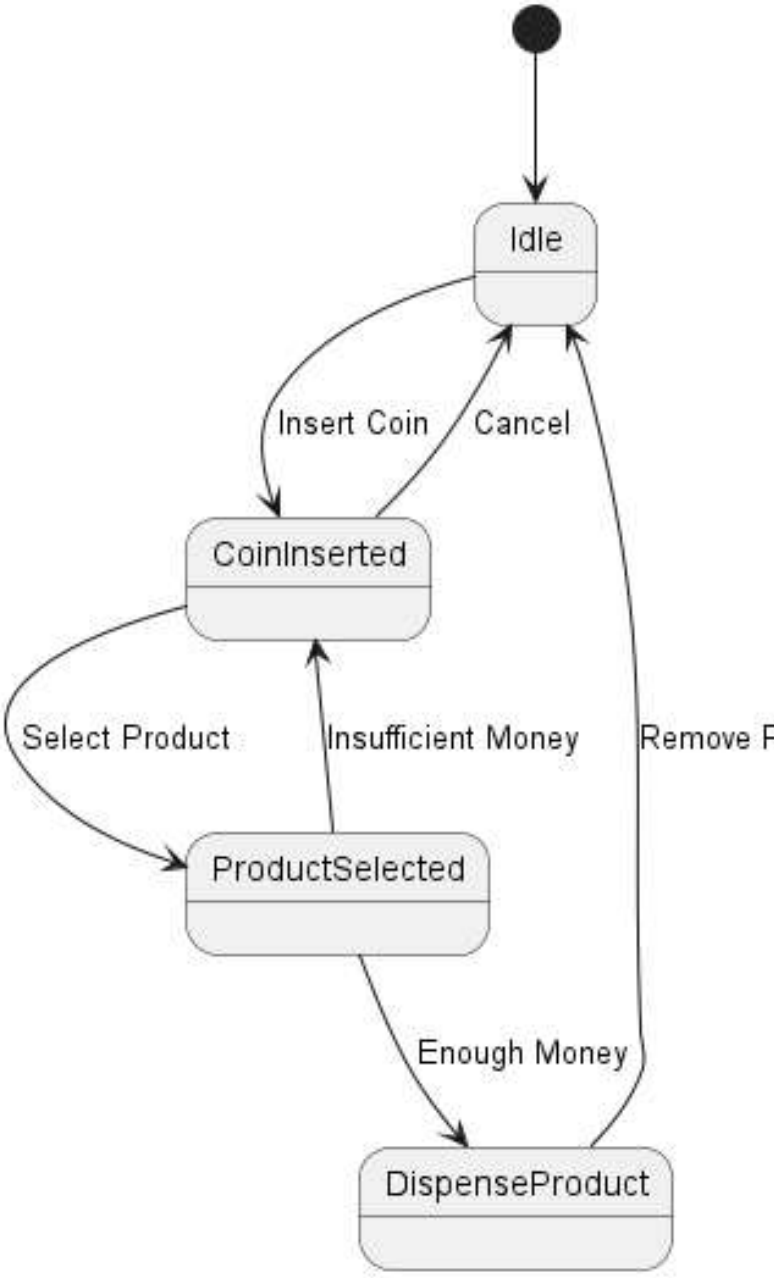
# State Diagrams

- A graphical representation of the states and transitions of object
- Models the dynamic behavior of a single object over time
- Captures the possible states and events that trigger state changes
- Useful for understanding complex object life cycles

# Key Eléments of State Diagrams

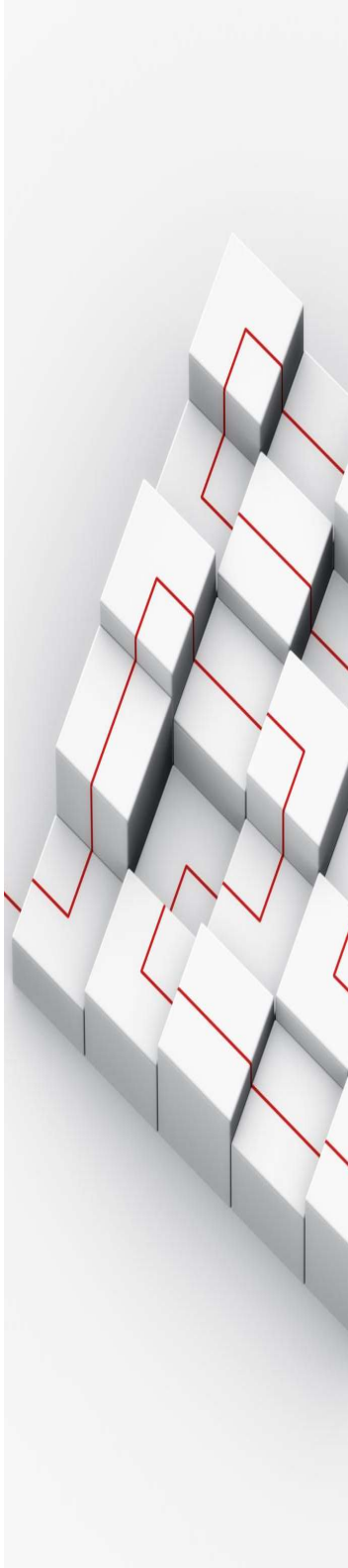
1. State: Represents a distinct condition or stage i
   object's life cycle
2. Transition: Arrows indicating the movement fro
   state to another
3. Event: A trigger that causes a transition betwee
4. Initial State: The starting state of the object
5. Final State: The ending state of the object

# Example: Vending Machine State Diagram



# Component Diagram

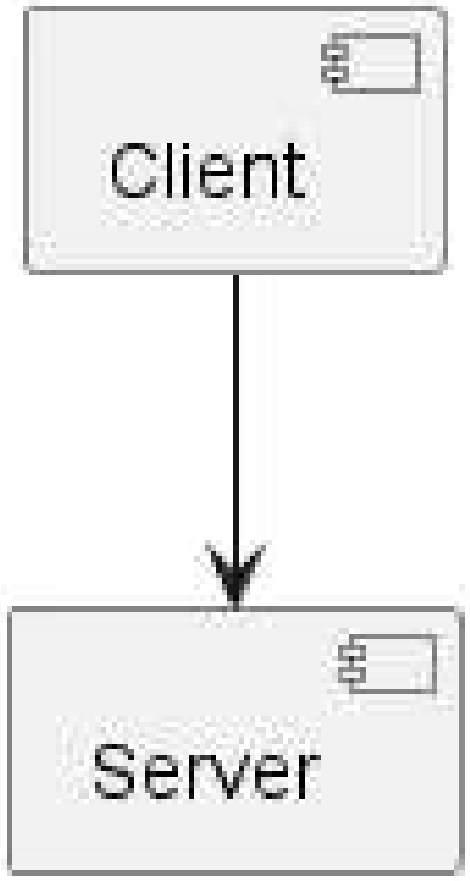
- Component diagrams are part of UML, which stands for Unified Modeling Language.
- They're used to show the organization and dependencies of the components in a software system.
- Components are modular parts of a system that encapsulate its functionality.
- They can represent physical entities such as executables, files, and libraries, or logical entities such as software modules and subsystems.



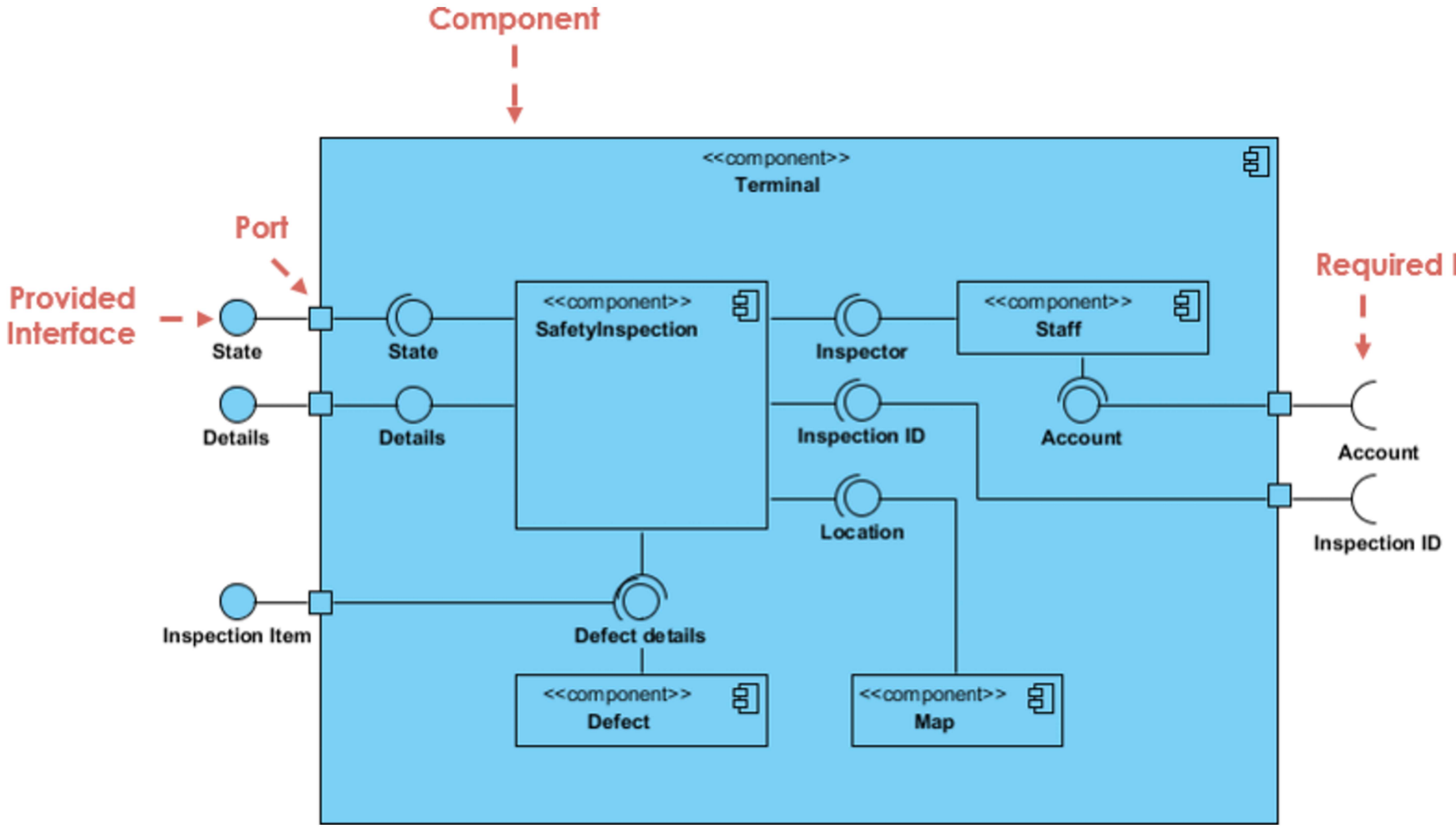
# Key Elements

- A component diagram is composed of components, interfaces, ports, and dependenci
- Components are depicted as rectangles with their name and optional stereotype in th and their implementation details in the lower part.
- Interfaces are depicted as circles with their name and optional stereotype in the cente
- Ports are depicted as small squares attached to a component or an interface, represer interaction between the two.
- Dependencies are depicted as arrows between components, indicating that one comp requires another component.

# Simple example

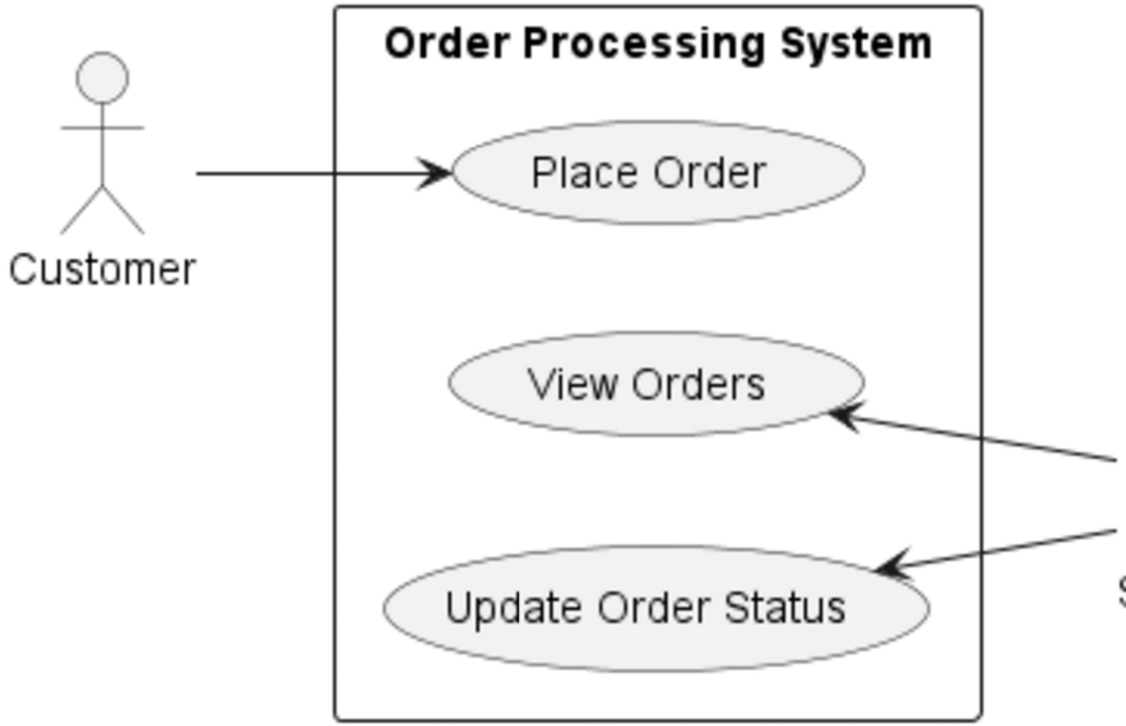


# Notation

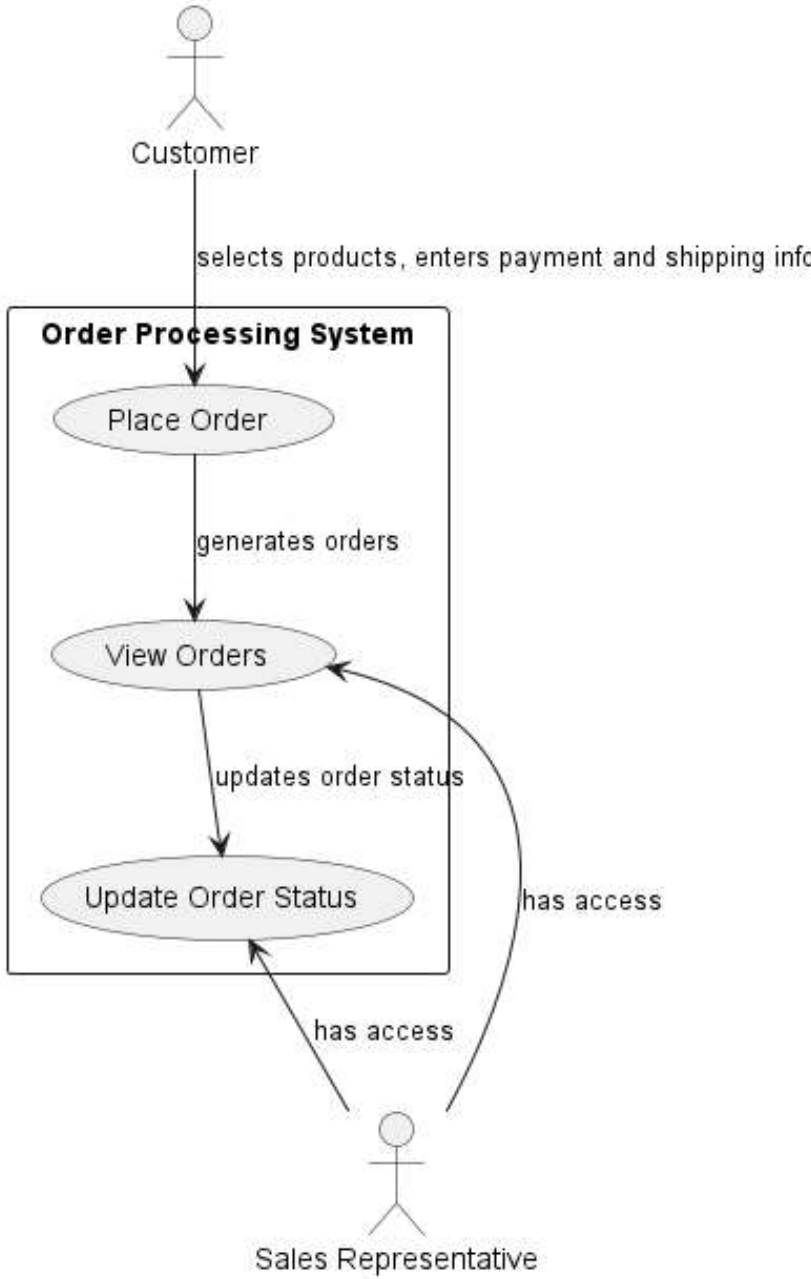

Component
<<component>>
Terminal
Port
Provided Interface
State
State
<<component>>
SafetyInspection
Inspector
<<component>>
Staff
Details
Details
Inspection ID
Account
Account
Location
Inspection ID
Inspection Item
Defect details
<<component>>
Defect
<<component>>
Map

# Use Case Diagrams

- They're used to capture the functional requirements of a system from the user's perspective.
- Use cases represent the interactions between the system and its users or other systems.
- Actors are the users or other systems that interact with the system.



# Example with relationships



# Deployment Diagram

- They're used to show how a system's software component and hardware resources are deployed or distributed.
- Nodes represent the system's physical or virtual computin resources, such as servers, clients, and routers.
- Artifacts represent the software components that are deployed on the nodes.

# Key elements

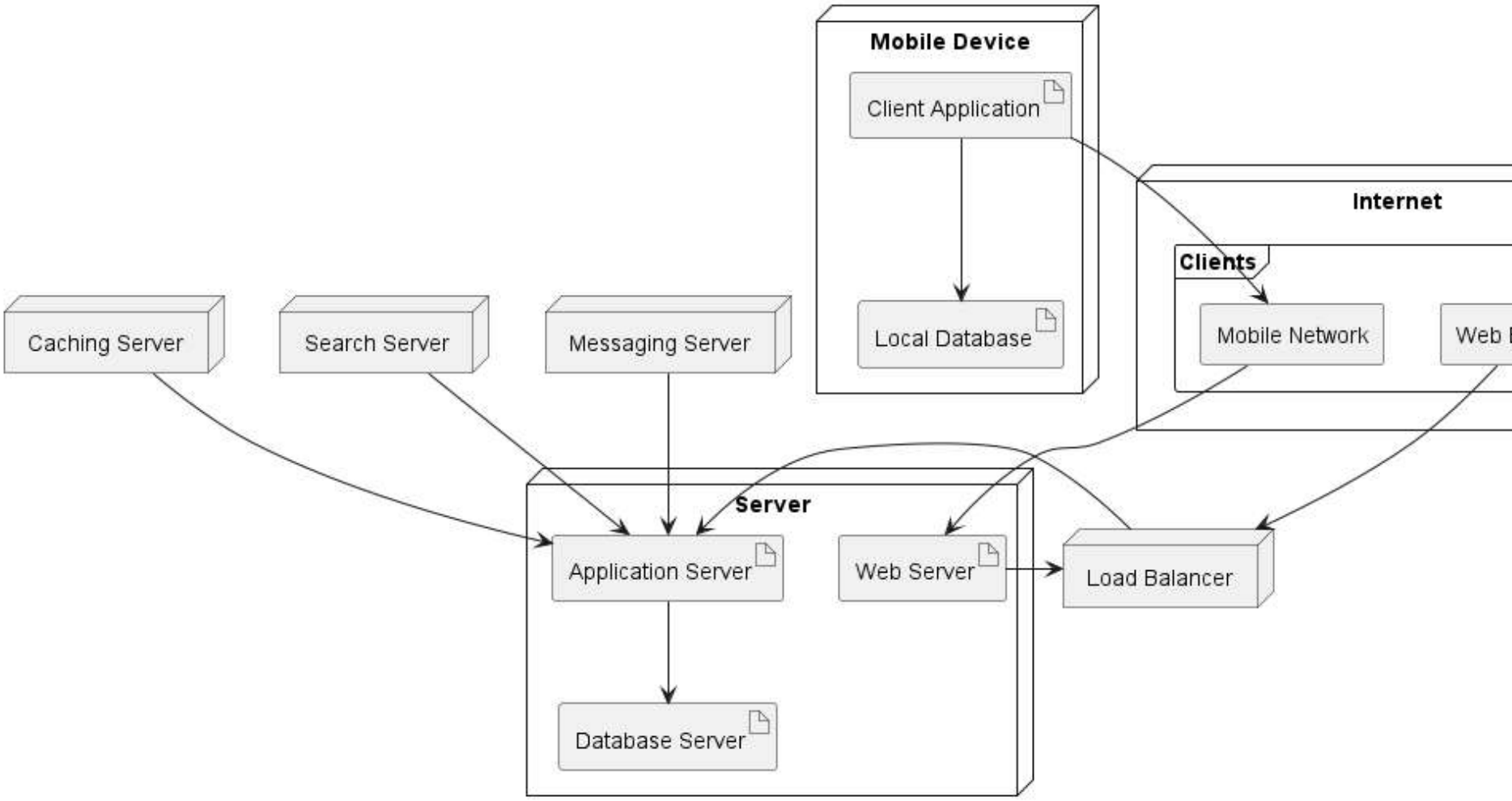
A deployment diagram is composed of nodes, artifacts, and relationships.

Nodes are depicted as boxes with their name and optional stereotype in the
and their implementation details in the lower part.

Artifacts are depicted as rectangles with their name and optional stereotyp

Relationships are depicted as arrows between nodes and artifacts, indicatin
deployment or communication between them.

# Example


Mobile Device
Client Application
Local Database
Internet
Clients
Mobile Network
Caching Server
Search Server
Messaging Server
Server
Application Server
Web Server
Load Balancer
Database Server

# UML and analysis

UML can be used at each stage of the software construction

UML for Analysis

Simplified Diagram

Avoid implementation decisions

Class Diagram as Entity Diagram

Represent domain objects and their attri

Sequence Diagram

Represent exchange between the system
external systems

Activity Diagram

Represent high level Business Processes


# UML for software design

This will be in another course

# Sources

- Cours de Martine Gautier
- Cours de François Charoy
- Cours de Pascal Molli
- Slides de Lou Franco
- http://www.extremeprogramming.org/
- UML Distilled: A Brief Guide to the Standard g Language, 3rd Edition
  - Martin Fowler
- http://www.uml.org/
  - Le site de référence
- De nombreuses figures sont issues de la spéc
- D'autres figures ont été générées en plantUM