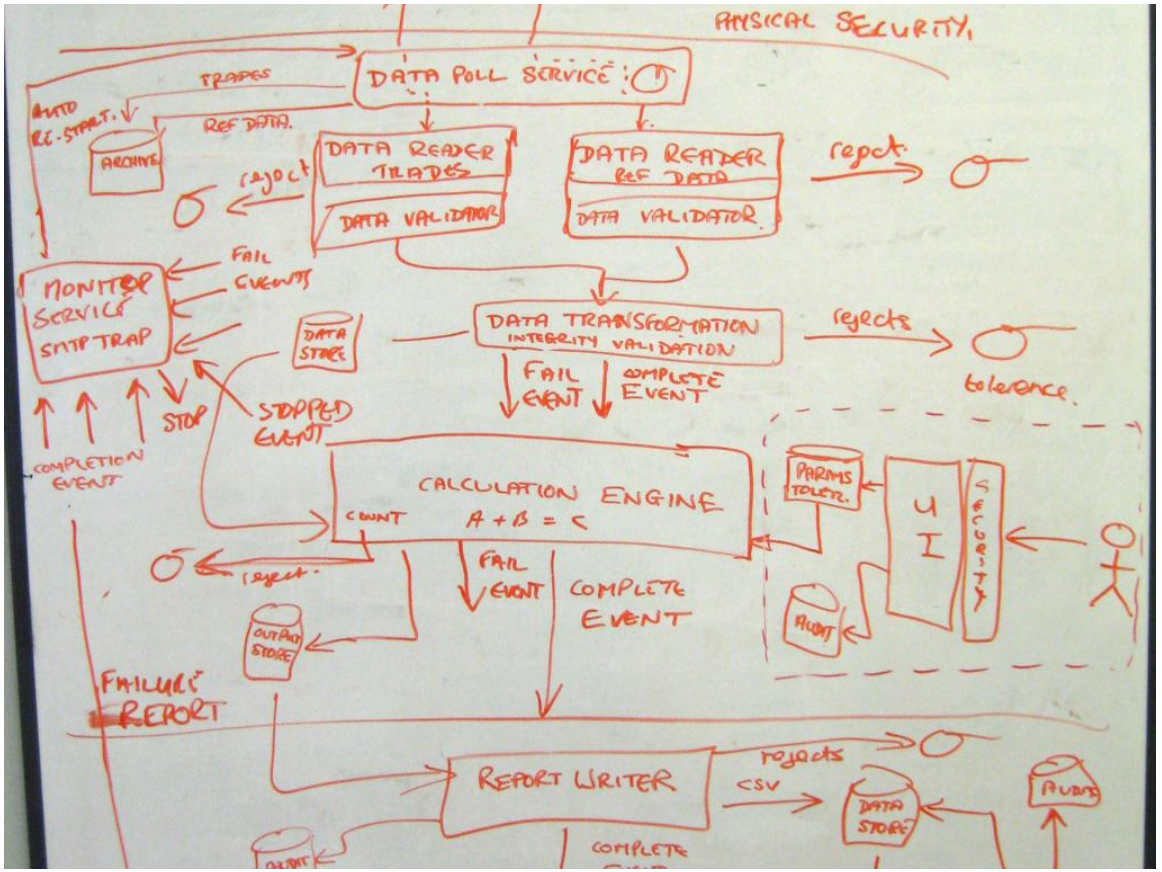
What is Software Architecture?

Architectural Viewpoints

What is software architecture ?
What is not software architecture ?

?

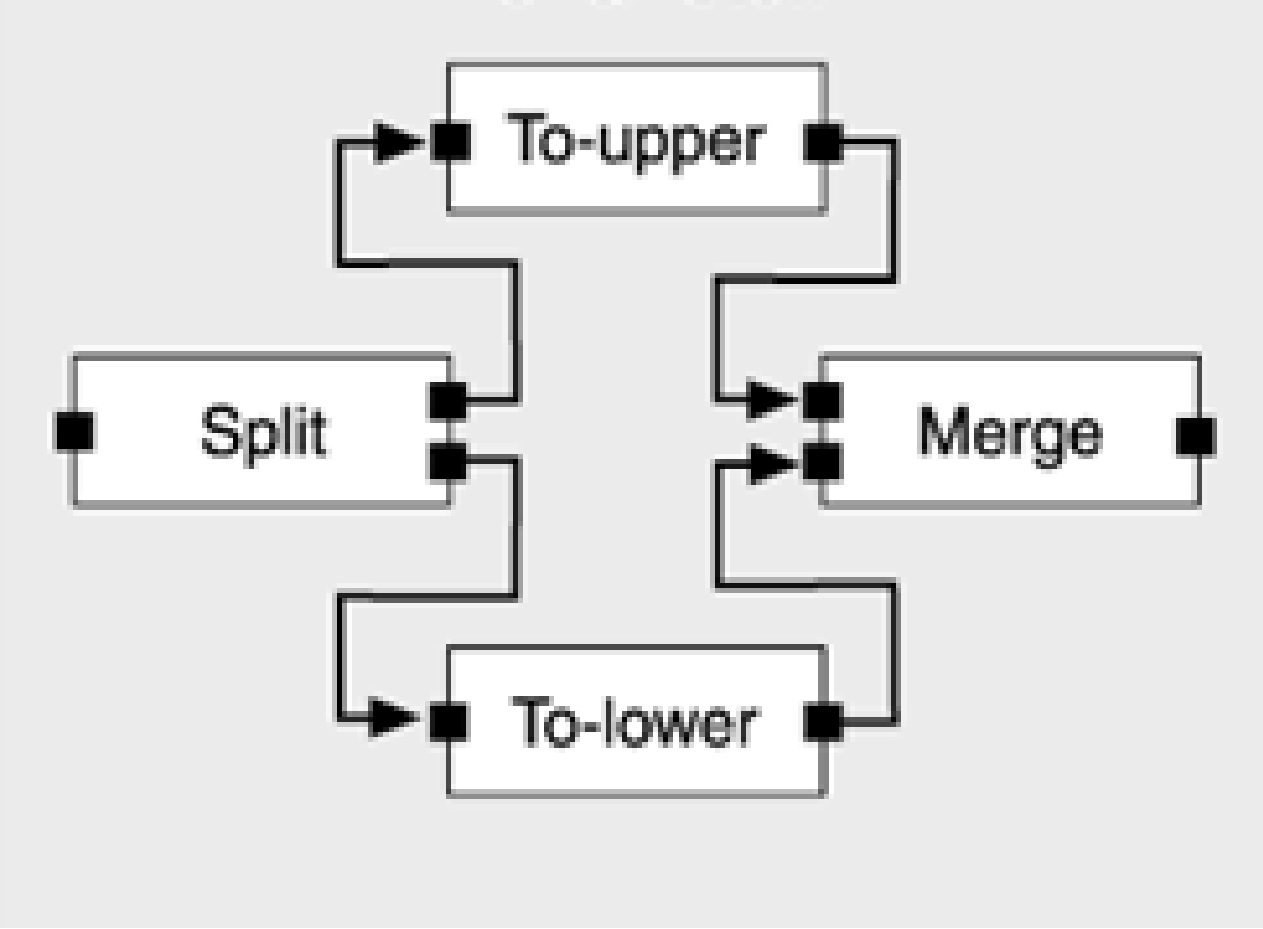
Very often, when speaking about software architecture, people are drawing “box and lines diagrams” ….



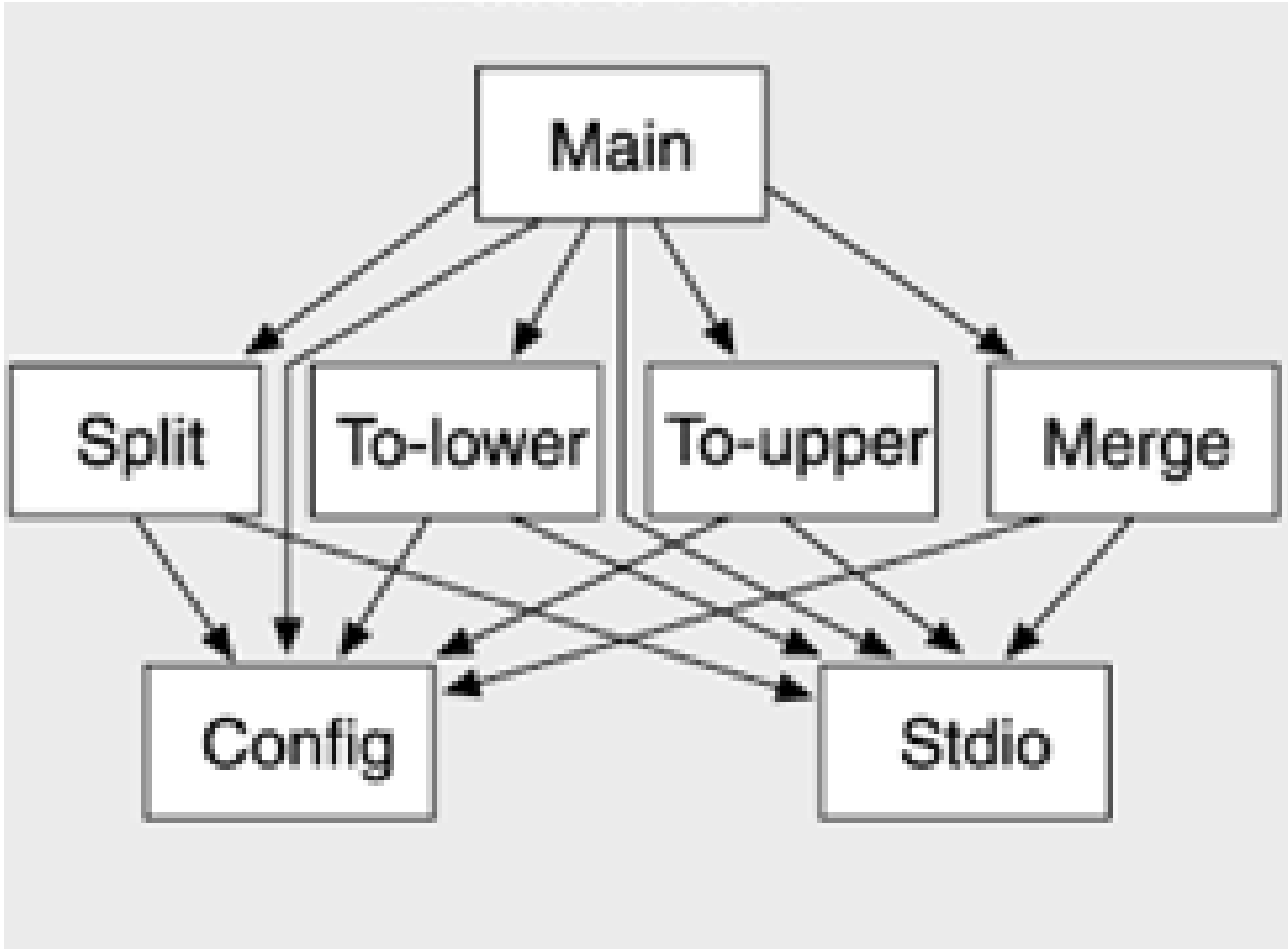
Example

- Case study: two software engineers developed together a system doing a simple text processing: the lines of the original text are transformed in following way: odd lines are rewritten in upper case and even lines are rewritten in lower case.
- The two engineers are asked, independently, to explain the architecture of their system.
- Each of them draws a diagram explaining the “architecture” of the system …

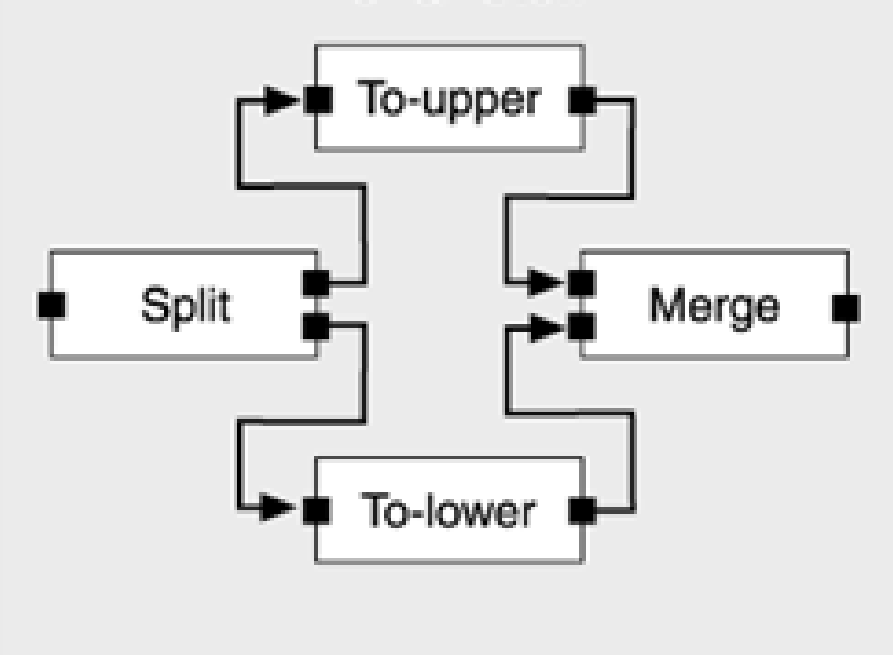
Example – Diagram by first engineer

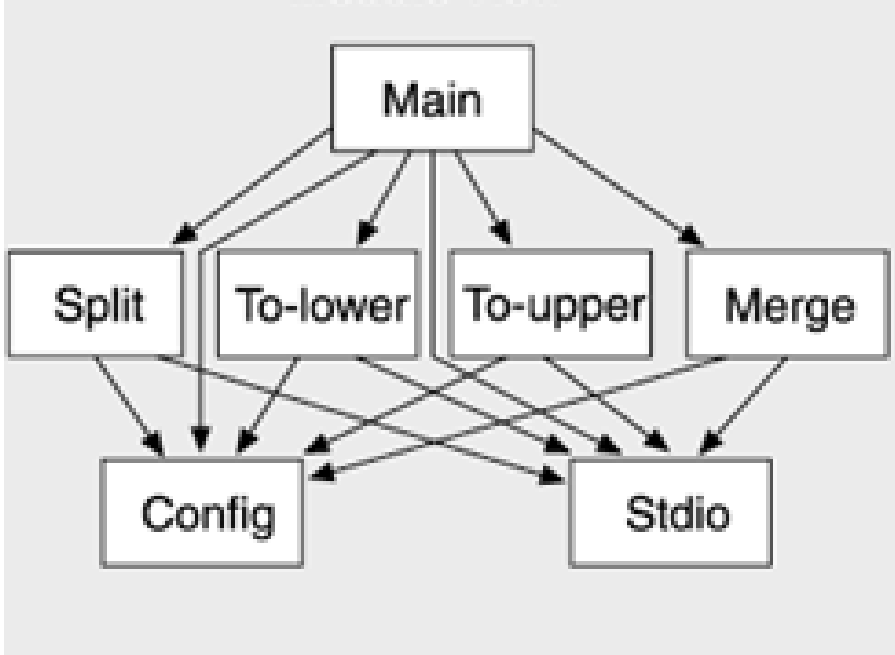


Example – Diagram by second engineer



?


To-upper
Split
Merge
To-lower


Main
Split
To-lower
To-upper
Merge
Config
Stdio

Software architecture - definition

- “The software architecture is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements and the relationships among them”

  (Bass et al)

Software architecture definition explained (1)

"The software architecture is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements and the relationships among them"
(Bass et al)

- Software elements: (the term *of architectural element* is preferred instead of architectural *component.*)
  - A software element is an **abstraction** of a **part** of the system
  - Architecture is an **abstraction** of a system
  - Abstractions are hiding certain details
  - A structure of a system is determined by elements and their relationships

Software architecture definition explained (2)

"The software architecture is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements and the relationships among them"
(Bass et al)

- Structures of the system: Architecture is described by **multiple** different structures (views):
  - The modules structure - the static structure of a system, results in the design phase.
  - The interactions structure – the dynamic structure that can be observed at runtime
  - The allocation/deployment structures – describes how software elements are mapped onto non-software elements (hardware, staff)

Software architecture definition explained (3)

"The software architecture is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements and the relationships among them"
(Bass et al)

- Inside each structural view, software elements and relationships among them may have different meanings:
  - **Elements** can be: objects, classes, functions, processes, programs, libraries, databases, etc.
  - **Relationships** between elements: part of, synchronization, function call, etc.

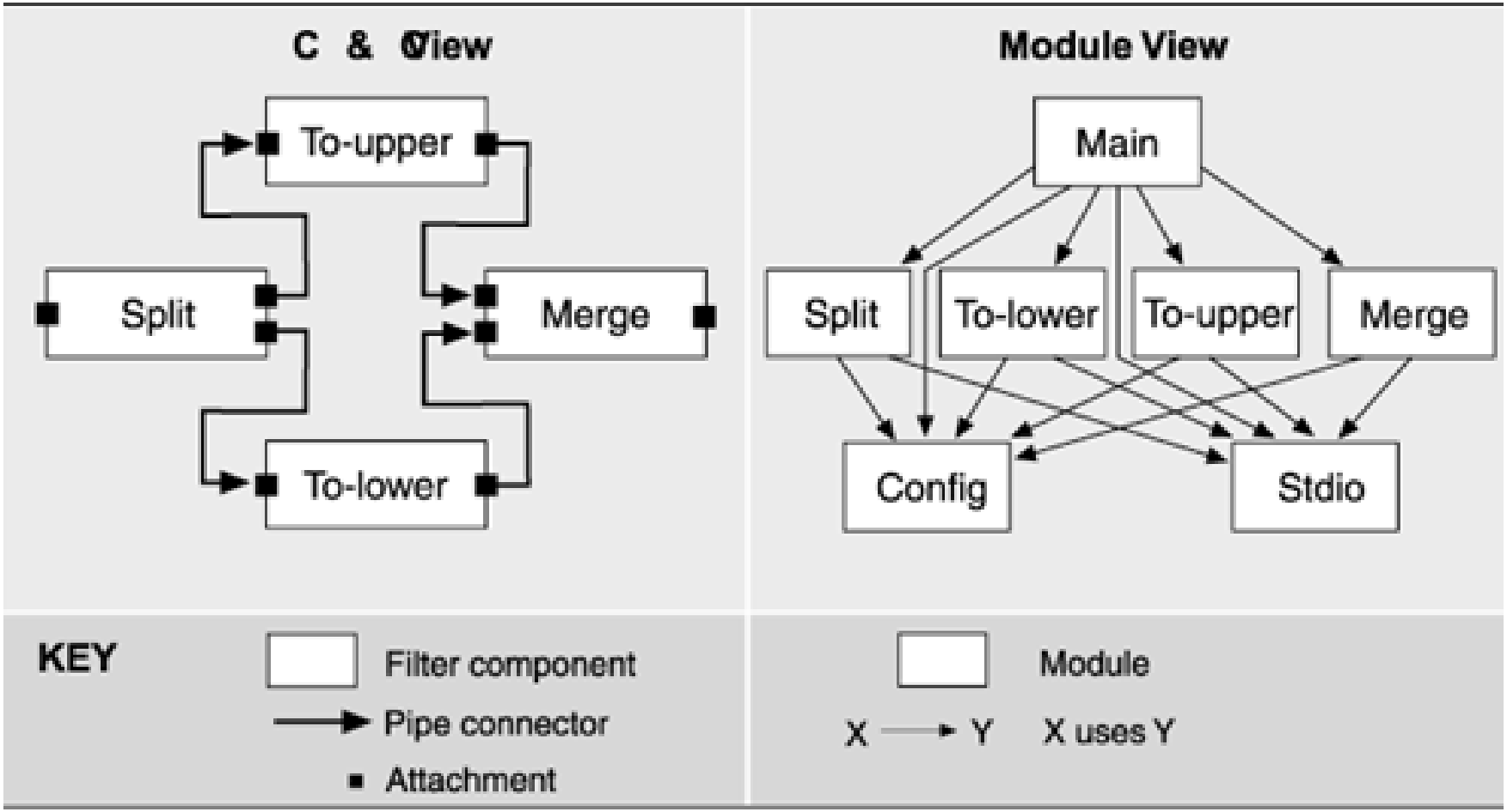
Multiple Architectural Structures = Architectural Viewpoints

- Software Architecture = a set of different views of different viewpoints
- A ***very* particular case**: UML for OO design:
  - Class Diagrams vs Object (Collaboration) Diagrams
- Similar:
  - Maps in geography: for the same territory, we have several different viewpoints: physical map, political map, economical or resource map, road network map, etc.

Multiple Architectural Structures = Architectural Viewpoints

- Viewpoints examples:
  - **Module** (Design/Functional Structure)
    - Elements: Units of implementation(classes, modules, packages, etc).
    - Relationships: depends (uses), is part of
    - Useful for: blueprint of design; impact analysis, etc
  - **Component and Connector** (Runtime)
    - Elements: Processing units or data storage units (objects, processes, servers, databases, etc)
    - Relationships: messages, communication, dataflows
    - Useful for: analysis of runtime properties: dataflows; replication of some parts, etc
  - **Allocation** (Deployment, Physical Structure, Team Structure)
    - Elements: Software and non-software elements
    - Relationships: allocated to
  - New viewpoints can be defined and used as needed

Module viewpoint vs C&C viewpoint



[Clements]

Describing software architectures

- IEEE/IEC/ISO 42010:2011 “Systems and software engineering – Architecture description”
- Revised by IEEE/IEC/ISO 42010:2022 “Software, systems and Enterprise – Architecture Description”
- It specifies the *requirements* for architectural descriptions
- It does not specify any description language
  - ADL’s (Architecture Description Languages): Wright, ACME, SysML, Rapide, etc
  - UML (component diagrams)

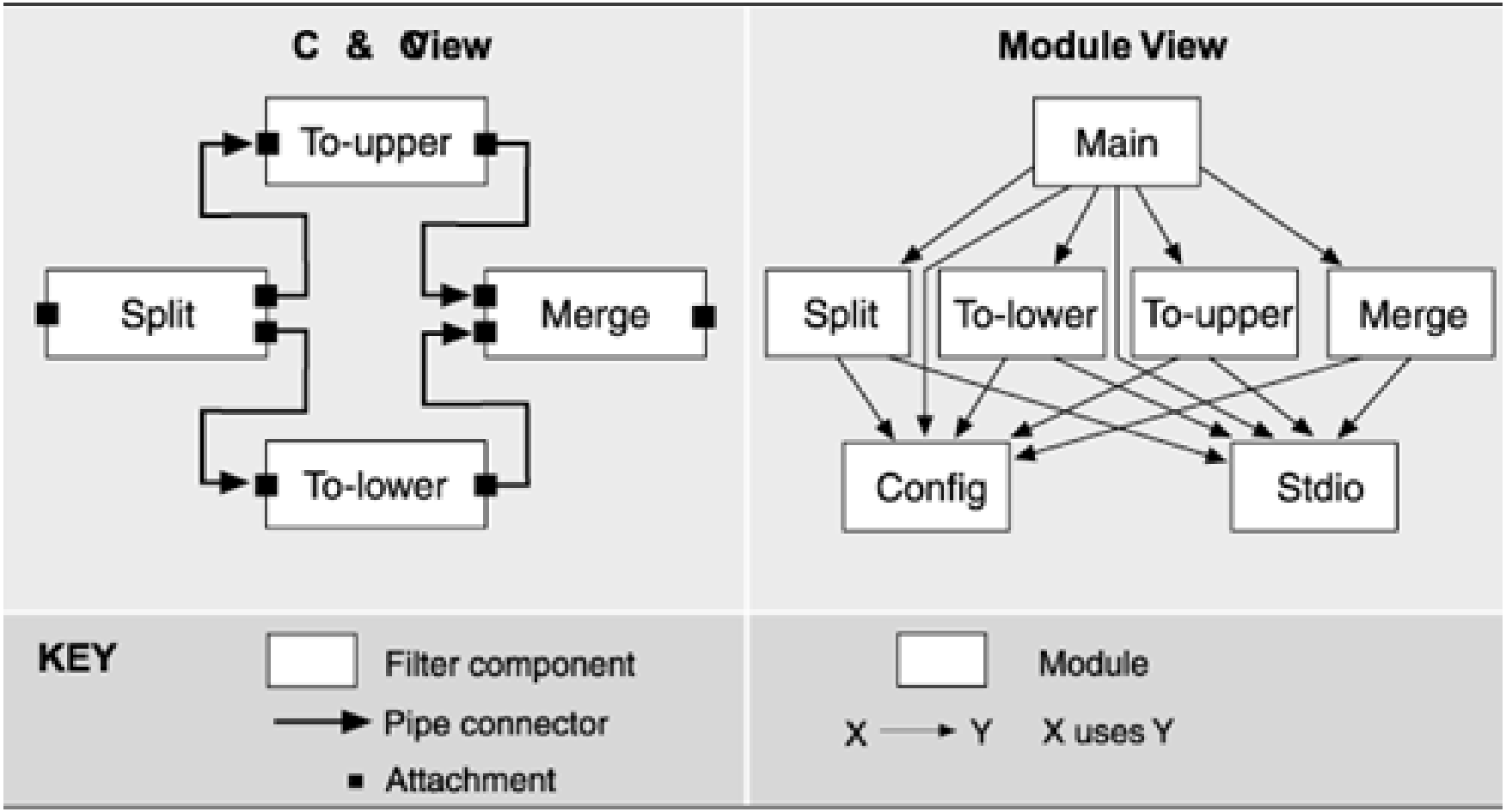
The key principles of the Architecture Description Standard

- Architecture descriptions should demonstrate how an architecture meets the needs of the system's diverse stakeholders;
- The architecture concerns of the diverse stakeholders can be addressed by an architecture description constructed with multiple **architecture views** of the system, where each view covers a subset of those concerns;
- The rules for well-formedness, completeness and analyzability of each architecture view should be explicit via an **architecture viewpoint**;

Box and lines diagrams?

- After all, software architectures could be represented by a **set** of *boxes and lines* diagrams, **if:**
  - for every diagram, you specify which **viewpoint** it covers and what represent the boxes(which types of elements) and what represent the lines (which types of relationships or connectors)

Example - the architecture

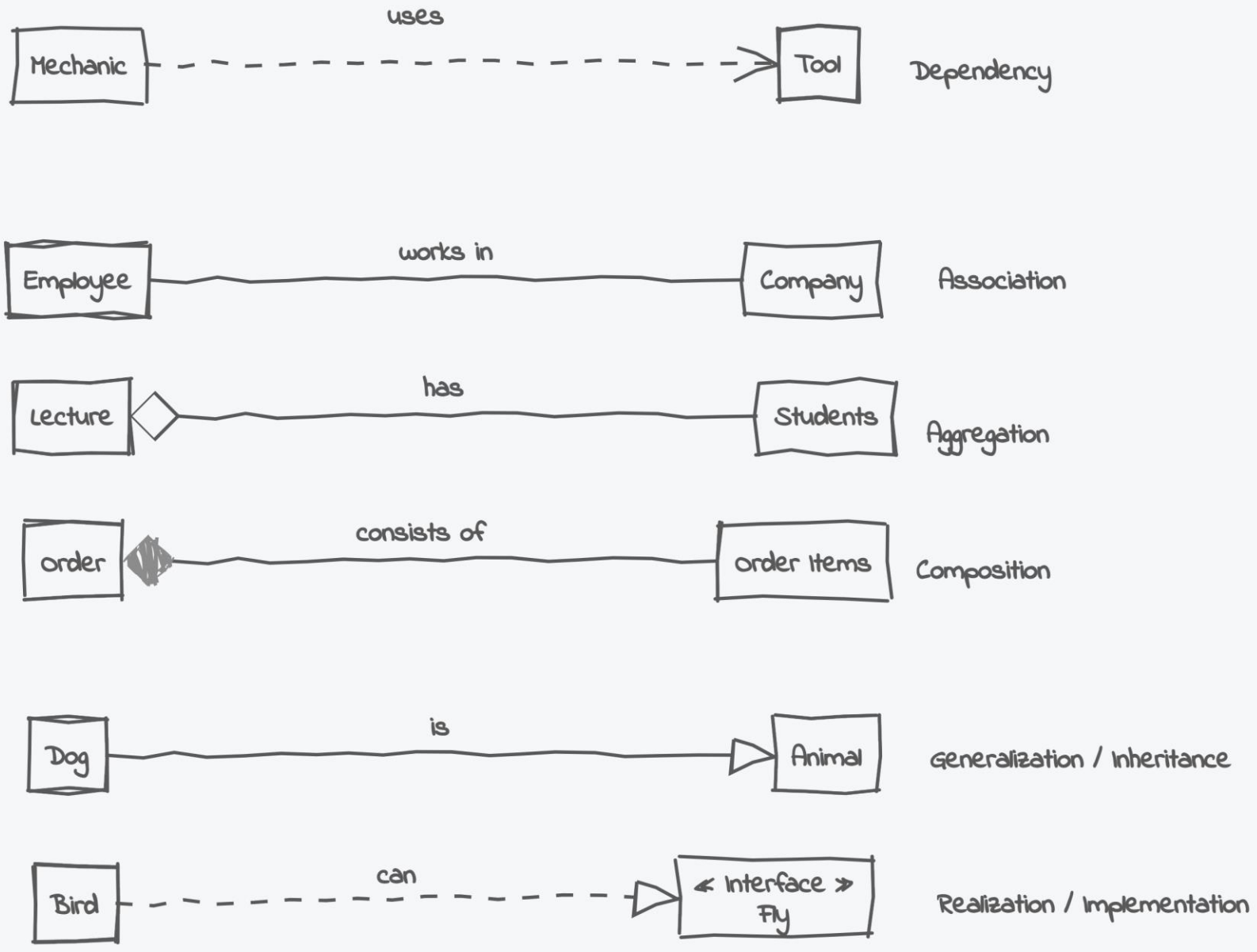


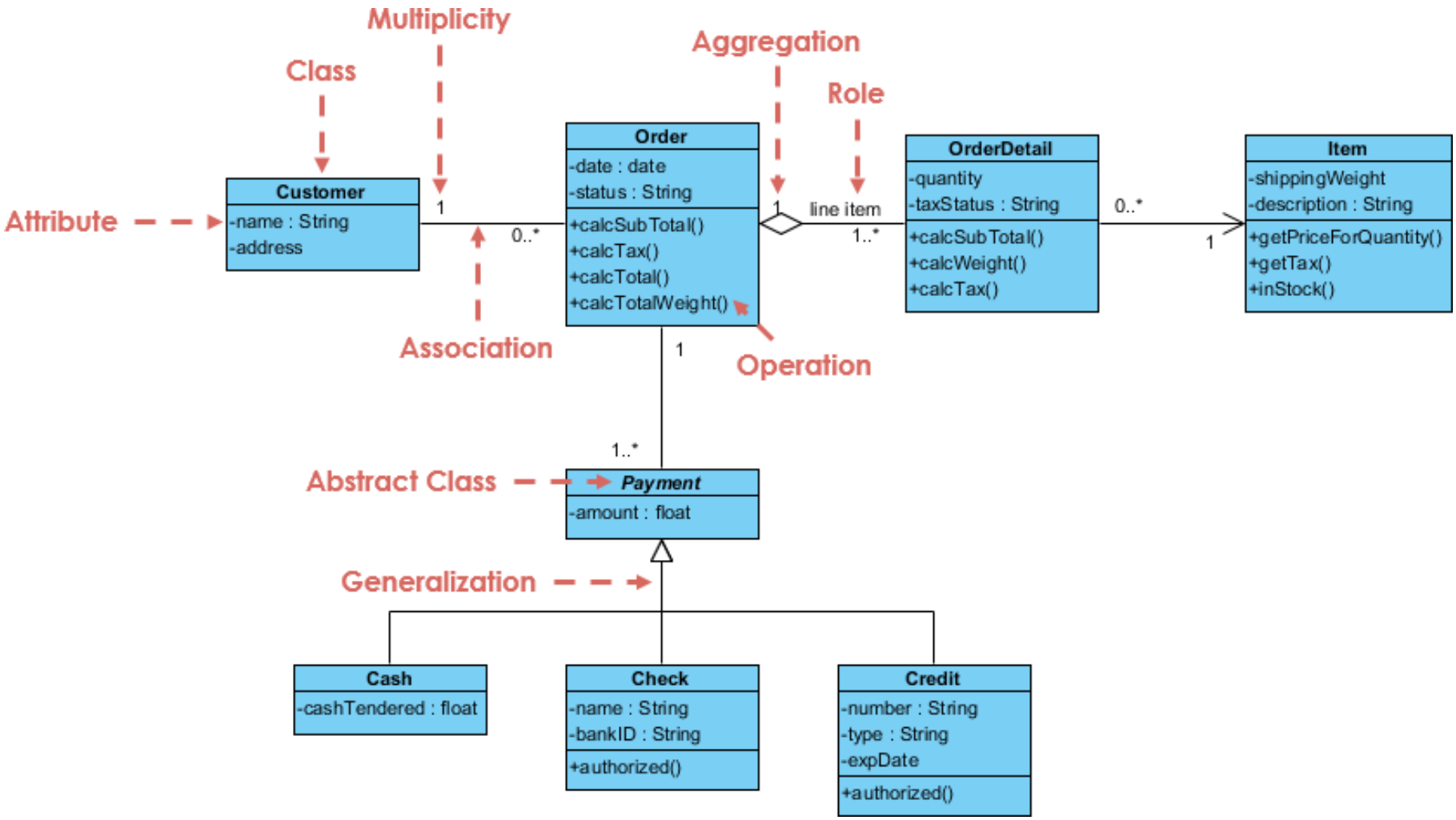
Review: UML (Unified Modelling Language)

- A standardized modelling language
- Used to specify and document software systems
- Used in object-oriented design
- UML includes:
  - Structural diagrams (ex: Class Diagram)
  - Behavioural diagrams (ex: Sequence Diagram)

UML Class Diagrams

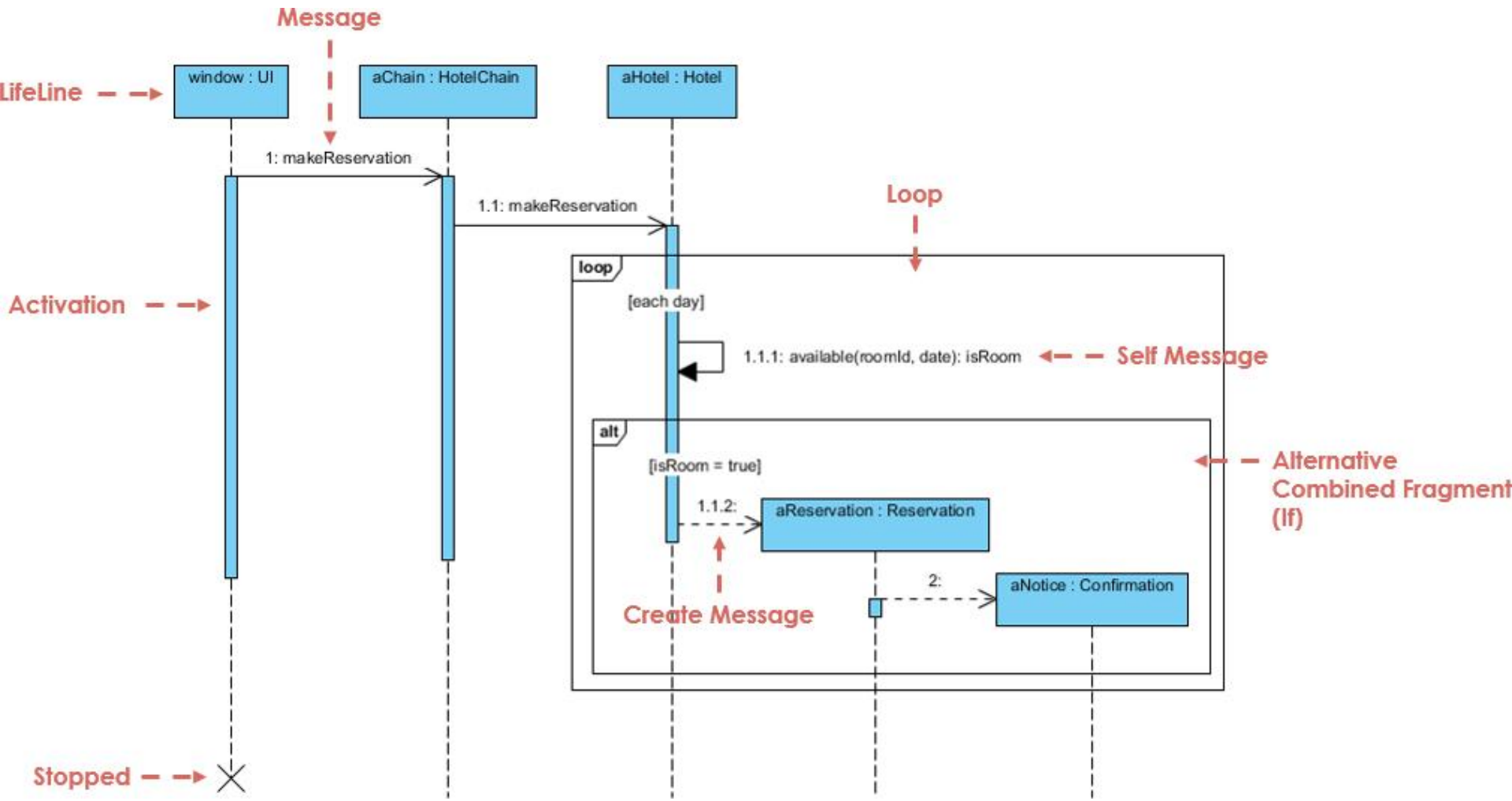
- Describes the **static structure** of a system
- Shows classes, attributes, methods, and relationships
- Main types of relationships:
  - Association
  - Aggregation
  - Composition
  - Inheritance (Generalization)
  - Dependency


uses
Mechanic
Tool
Dependency
works in
Employee
Company
Association
has
Lecture
Students
Aggregation
consists of
order
order items
Composition
is
Dog
Animal
Generalization / Inheritance
can
Bird
« Interface »
Fly
Realization / Implementation


Multiplicity
Class
Aggregation
Role
Attribute
Customer
-name : String
-address
1
0..*
Order
-date : date
-status : String
+calcSubTotal()
+calcTax()
+calcTotal()
+calcTotalWeight()
1
line item
1..*
OrderDetail
-quantity
-taxStatus : String
+calcSubTotal()
+calcWeight()
+calcTax()
0..*
1
Item
-shippingWeight
-description : String
+getPriceForQuantity()
+getTax()
+inStock()
Association
Operation
1
1..*
Abstract Class
Payment
-amount : float
Generalization
Cash
-cashTendered : float
Check
-name : String
-bankID : String
+authorized()
Credit
-number : String
-type : String
-expDate
+authorized()

UML Sequence Diagram

- Describes (part of) the **runtime behaviour** of a system
- Shows **interactions** between objects over time
- Emphasizes the order of messages
  - A message: object.method(parameters)


Message
LifeLine
window : UI
aChain : HotelChain
aHotel : Hotel
1: makeReservation
1.1: makeReservation
Loop
loop
[each day]
Activation
1.1.1: available(roomId, date): isRoom
Self Message
alt
[isRoom = true]
Alternative
Combined Fragment
(If)
1.1.2:
aReservation : Reservation
2:
aNotice : Confirmation
Create Message
Stopped