Upprifjun – UML ofl.

Software Requirements and Design
T-216-GHOH

Skúli Arnlaugsson | 4. nóvember 2019

HÁSKÓLINN Í REYKJAVÍK
REYKJAVÍK UNIVERSITY

What is UML

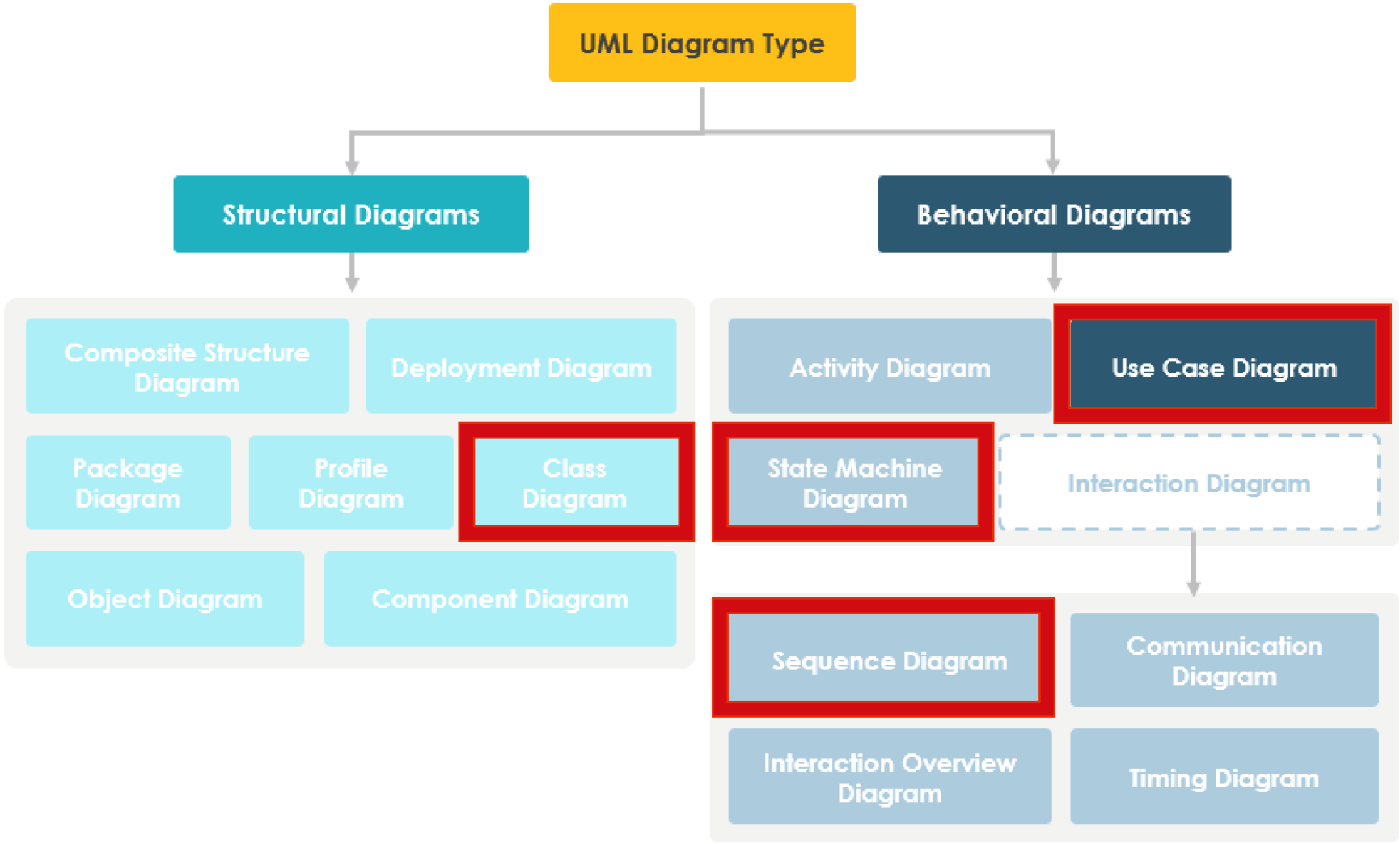
- A language used to *describe the behavior, functionality, and flow within a system* as well as *the system's interaction with the outside world*.
- A Modeling Language (mostly graphical)
  - Concepts (í. hugtök)
  - Notation (í. skrifháttur)
  - Organization (í. skipulag)

What UML is (cont.)

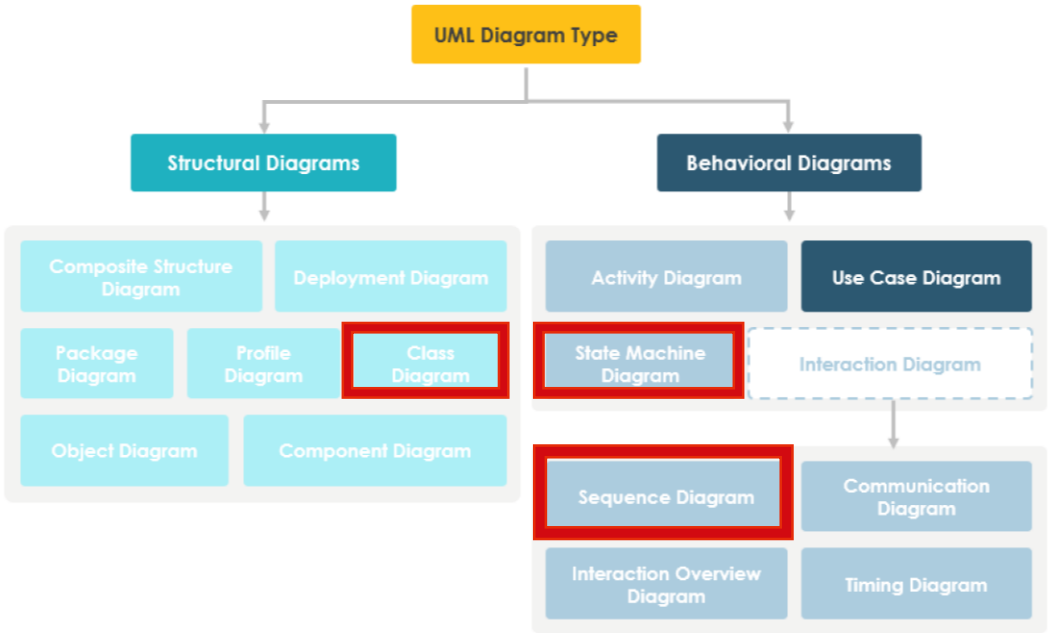
- *De facto* standard for modeling Software Systems
  - But not limited to only Software Systems
- Helps create clarity and understanding

What UML is (cont.)

- UML is big! 800+ pages of specification
- In this course we will learn about a small subset of the UML standard
- Including:
  - Use cases and partly Use Case Diagrams (í. notkunartilvik)
  - Class Diagrams (í. klasarit)
  - State Diagrams (í. stöðurit)
  - Sequence Diagrams (í. runurit)


UML Diagram Type
Structural Diagrams
Behavioral Diagrams
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Class Diagram
Object Diagram
Component Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Interaction Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram

Big picture: Our UML Diagrams



- The class diagram (ísl. klasarit) details our system's main objects, and how they are connected. The structure of our system.
- The state machine diagram (ísl. stöðurit) details the how the system or parts of the systems behave across multiple use cases.
- The sequence diagram shows a single use case and how objects (from our class diagram) interact with each other during the use case.

Possible steps in our example

Identify the nouns and refine the list

- Find texts from analysis phase to work with

Make a class diagram

- Entity classes first from the nouns, then identify UI, Business and Repository layer classes

Look at e.g. use cases, find those that seem to involve a lot of logic and steps

- These might be the ones that all parties need to agree upon

Make a sequence diagram of those to clear up possible logic/design decisions and get confidence that the design is right, before entering the programming phase

What is a use case?

- *A use case is a written description of some system functionality that creates value for a user of the system.*
- Unlike most of UML notation, **use cases are written** and not necessarily drawn.
- The focus should be on **user goals**, and they should describe the interaction of users with the system.
- **What the user needs to do**, not how the user does it.

What is a use case? (cont.)

What the user needs to do

- The user needs to authenticate themselves

How the user does it

- The user enters their email and password
- The user clicks on Facebook logo to login using their Facebook account

What the user needs to do

- The user needs to buy a product

How the user does it

- The user looks up the product on the website.
- The user adds the product to the cart.
- The user clicks checkout and pays for the product.

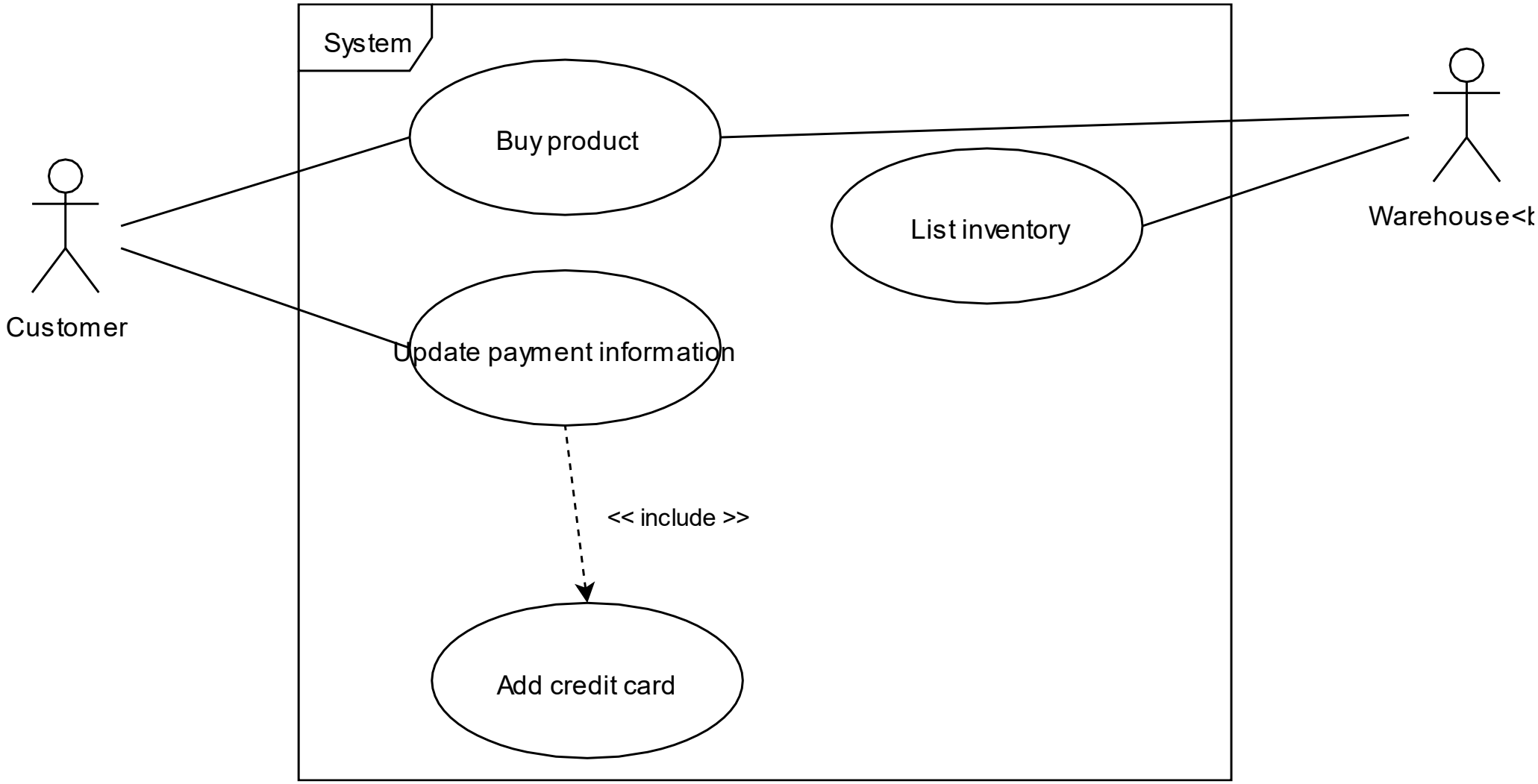
What is a use case? (cont.)

- A use case is a set of one or more *scenarios* tied together by a common user goal.
- A scenario represents one path through a use case.
- A scenario is a sequence of steps describing an interaction between a user and the system.

Use case format (empty)

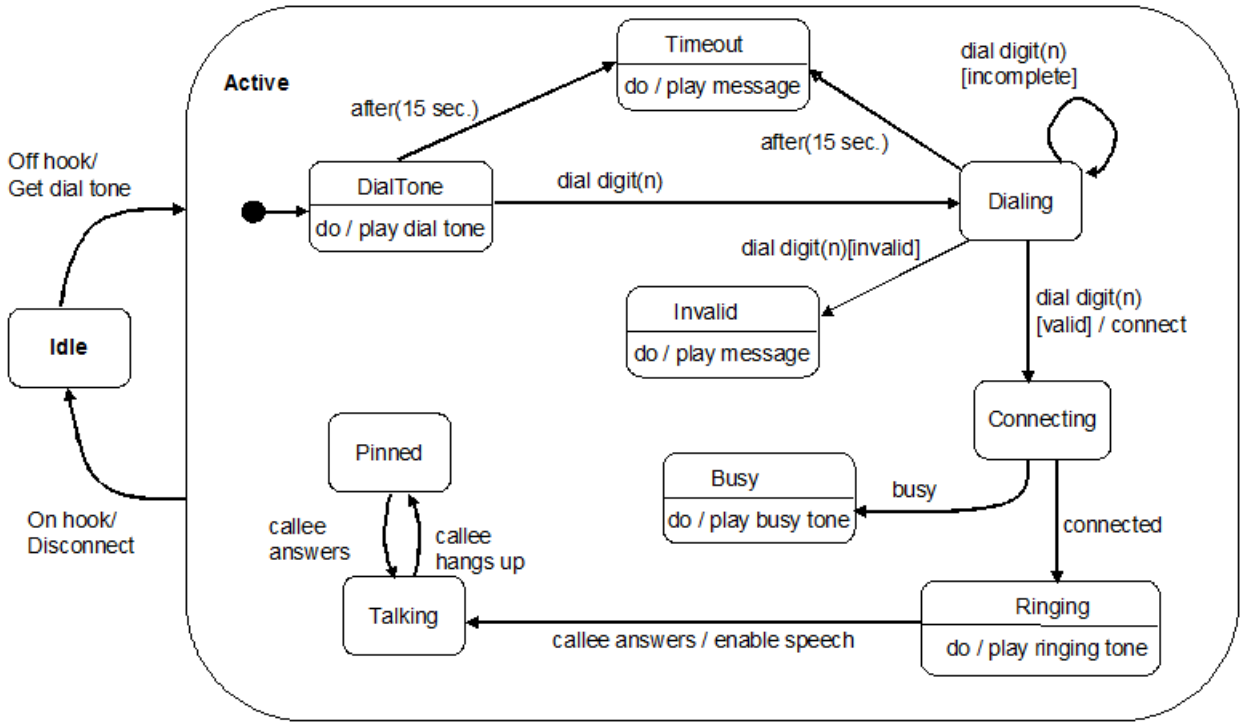
Name	
Number	
Description	
Priority	
Author	
Source	
Actors	
Precondition	
Postcondition	
Main success scenario	
Extensions	

Use case diagrams



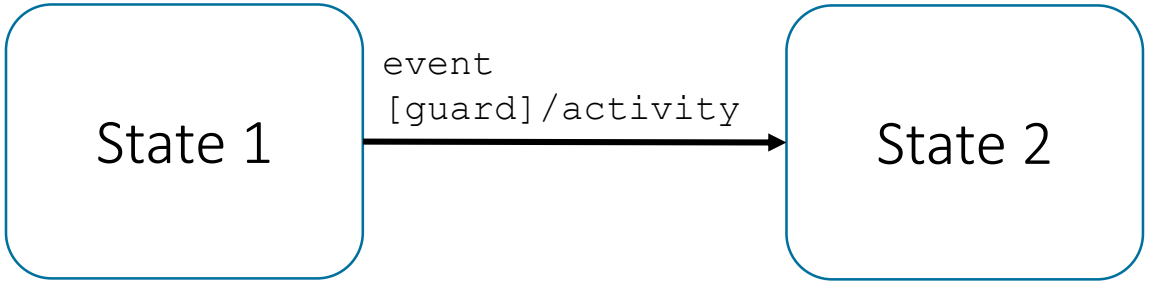
State Diagram (í. stöðurit)

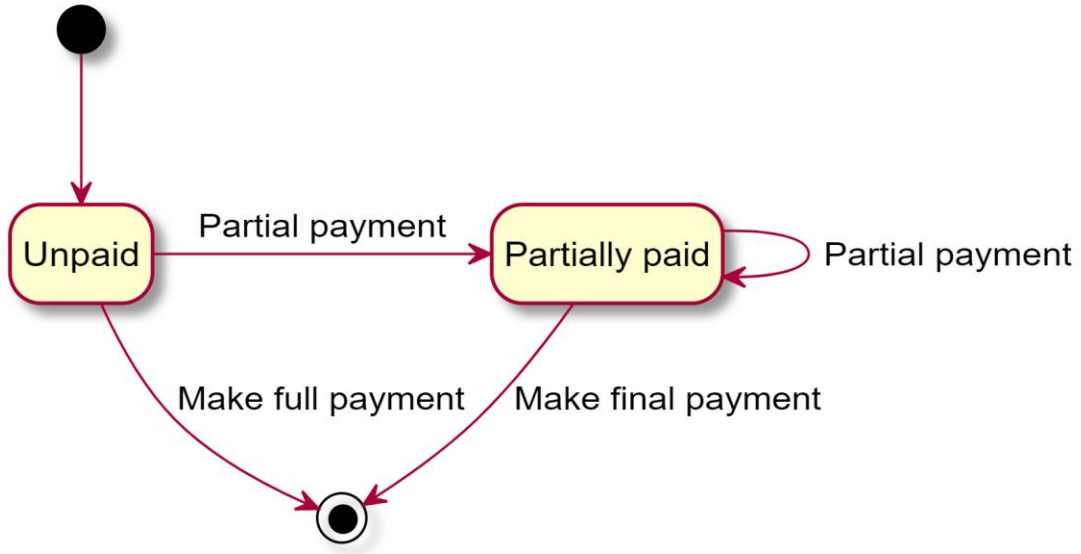
- State Machine Diagram
- Specify the behavior of some part of the system in terms of states and the transition between them
- Systems might behave differently depending on what state they are in, the State Diagram helps explain this behavior



State diagrams (contd).

- States
  - Initial state
  - Final state(s)
- Transitions
  - Guards
  - (Choice, fork, join)
- Internal actions (entry, exit, do, include)
- Sub states, compound state, composite state
- Concurrent states
- History states



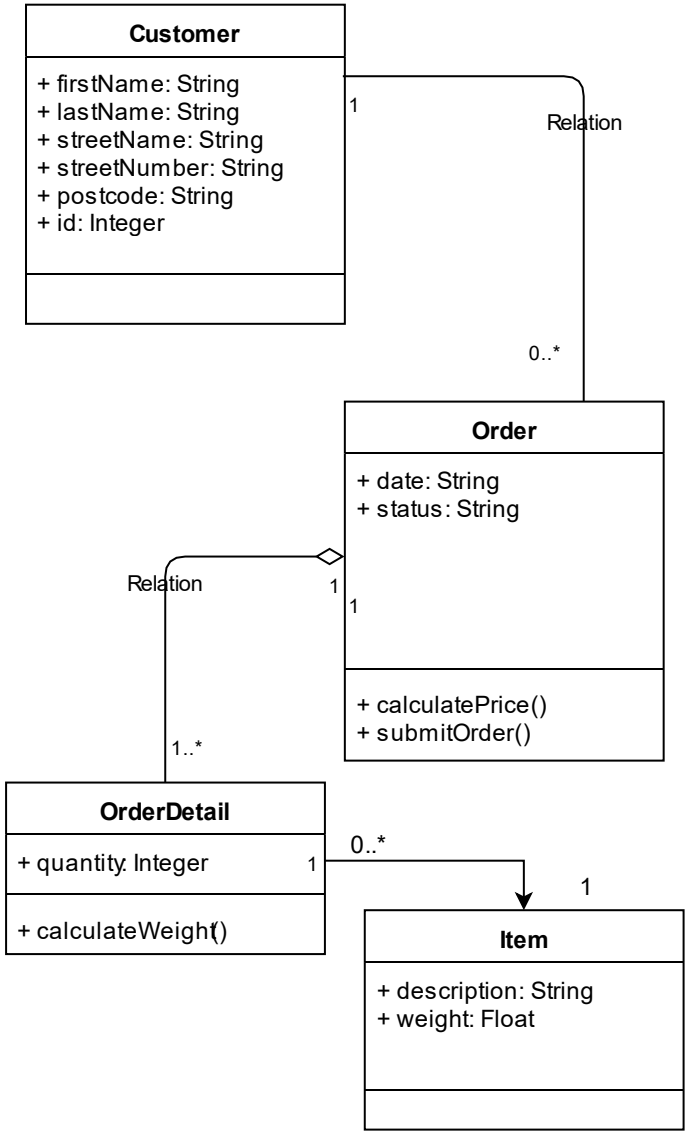


Noun identification (ísl. nafnorðagreining)

- **Step 1**: Use whatever text we have prepared in requirement gathering phase, e.g.:
  - general description of the system
  - list of requirements
  - *use cases*
- **Step 2**: Identify the nouns (*ísl. nafnorð*) in the text
- **Step 3**: Categorize the nouns into (usually) 3 categories:
  a) not relevant, can be thrown away
  b) relevant, are immediately recognizable as classes
  c) fuzzy, might be but we're not quite sure

Noun identification

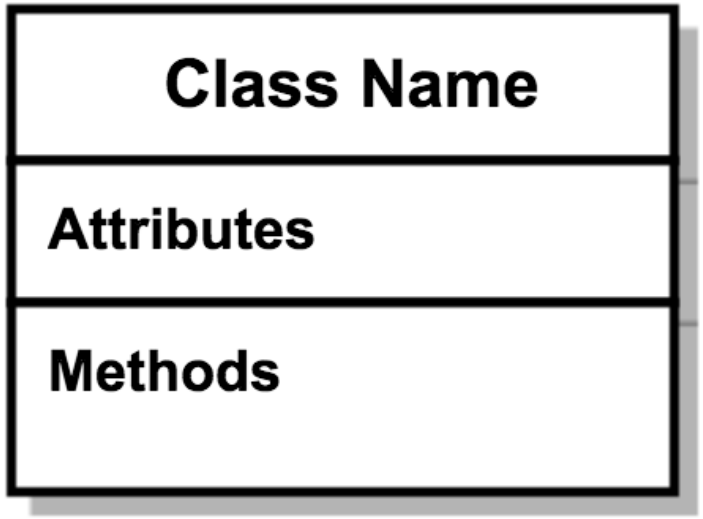
- **Step 4**: Refine the list:
  - remove **redundant** words, i.e. if two words have same or similar meanings
  - remove **vague** words, where the meaning is not exactly clear
  - remove **meta-words**, words like "system" and "data" (these would be terrible classes!)
  - remove words which are **outside the scope** of the project
  - remove words which will obviously **become attributes** (or even operations), but won't become classes



Class Diagrams (í. klasarit)

- Model the Terms that are used in the System
- Model collaborations between Classes
- Reason about the System and how it will work
- Provide a basis for implementation in an Object Oriented Language

Classes UML



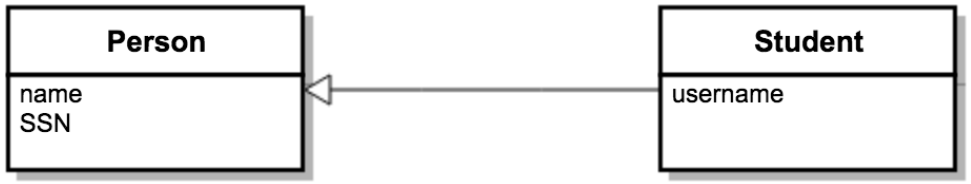
The UML representation of a class is a rectangle containing three compartments.

```
visability name: type multiplicity = default {property-string}
```

```
visability name(parameterList): return-type {property-string}
```

Class relationships

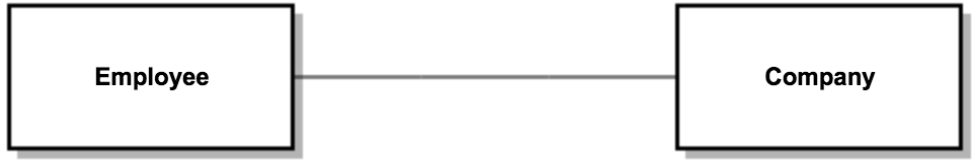
Inheritance



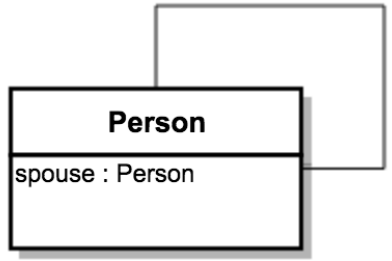
parent class/
super class

child class/
sub class

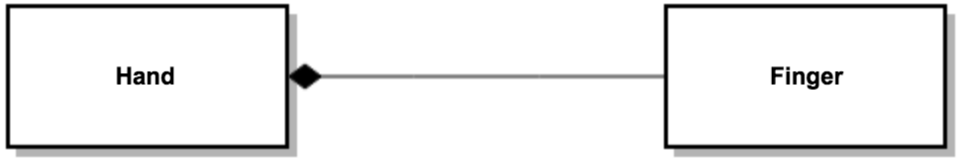
Association



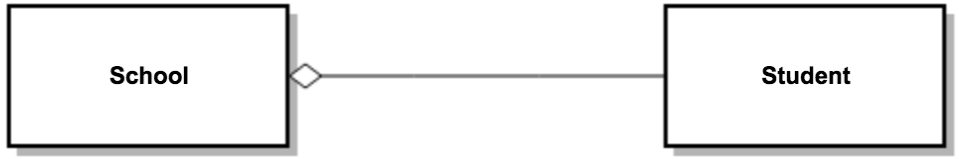
Association - reflexive



Composition



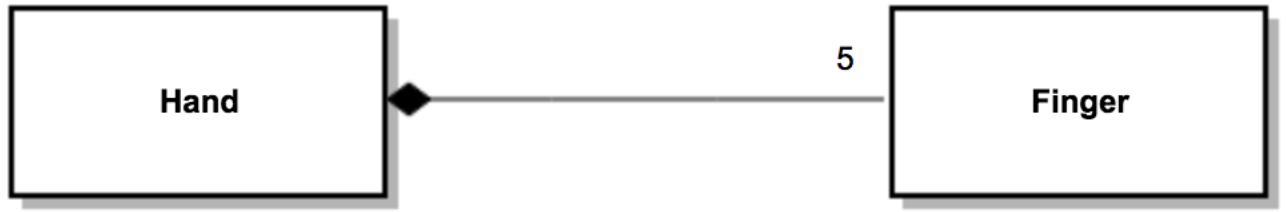
Aggregation



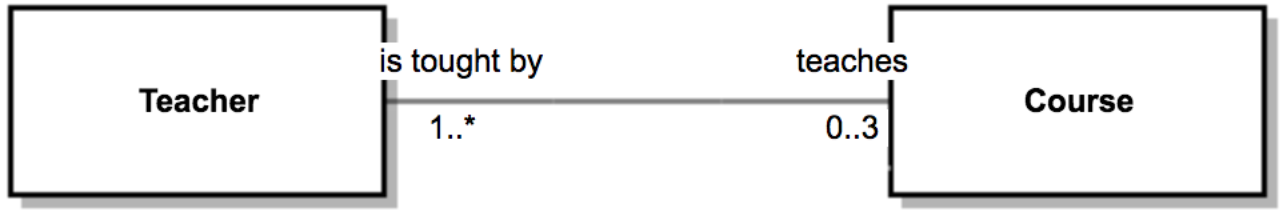
Dependency

Multipliers

- Multiplicity (*ísl. margfeldisþáttur*) defines how many instances of one class can be related to a **single instance** of another class
- Examples:



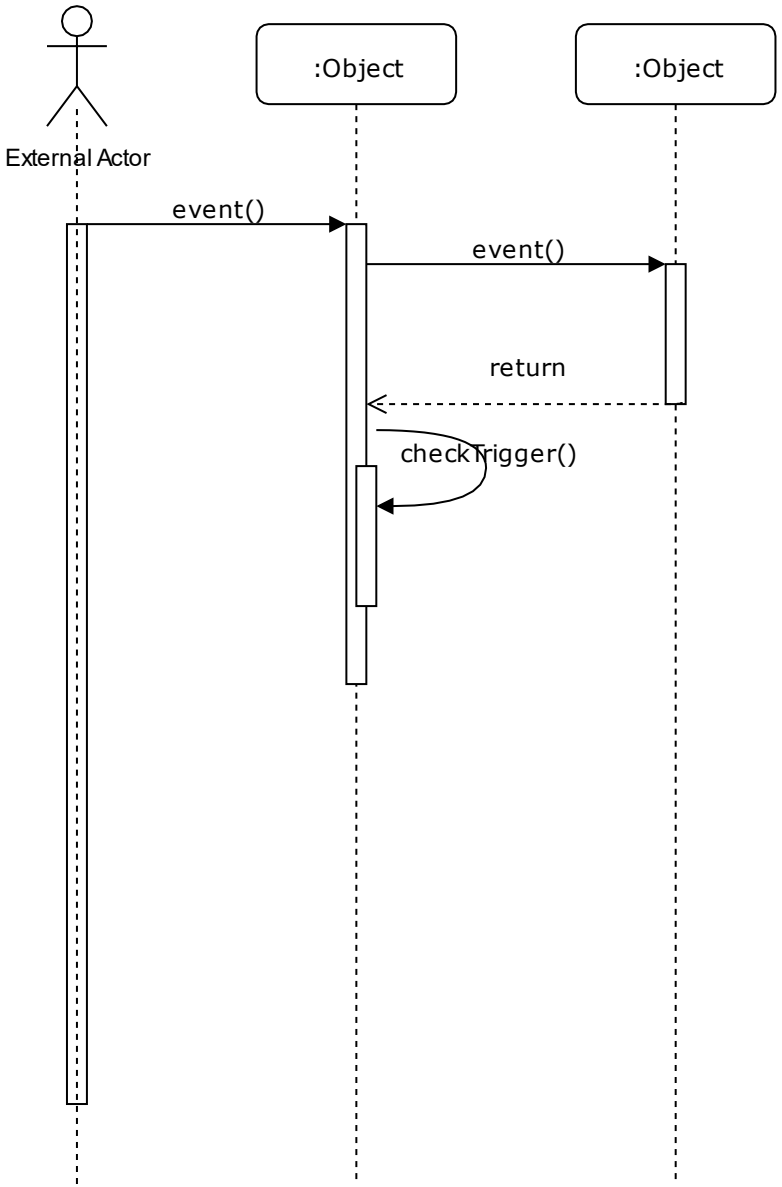
A hand has exactly 5 fingers



Different multipliers

- Sometimes the multiplier is just a
  `single number`
- Otherwise, they are in the format
  `[lower bound]..[upper bound]`
- If there are no restrictions to the number of class instances, the star (*) is used

*****	Any number (including zero and one)
0..*	Same as *
0..1	Either zero or 1
1	Exactly one
1..*	One or more
2..4	2,3 or 4

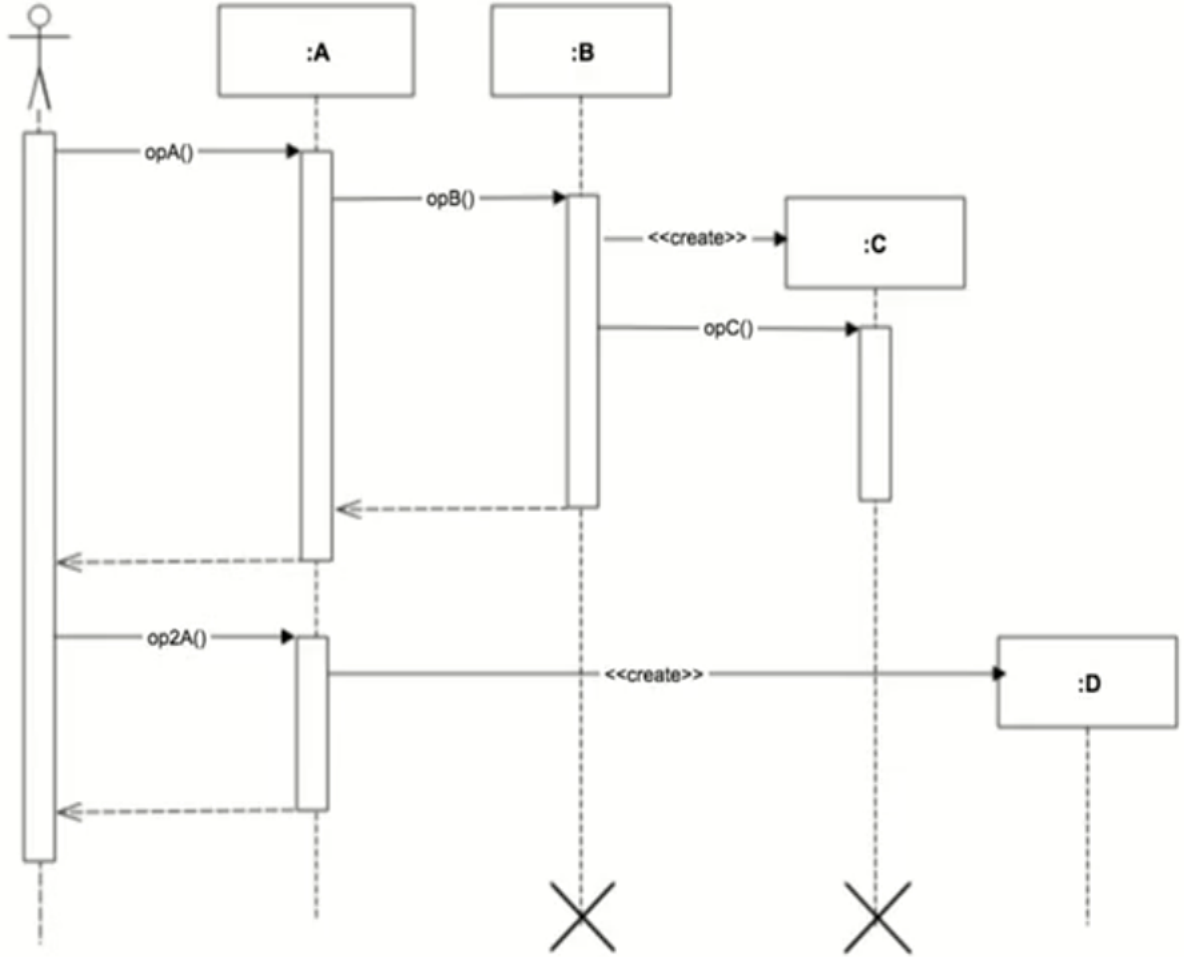


Sequence Diagrams (í. runurit)

- Interaction Sequence Diagram
- Provide a view of the Interaction between Entities of the Systems
- Supports Use Cases
- Models a Sequence of actions and interactions

Sequence diagrams topics

- Actors
- Lifelines
- Activity
- Operation vs. Message
  - synchronous vs. asynchronous
- Instance creation & deletion
- Reply
- Self calls
- Interaction frames
  - Alt, opt, par, loop, ref, …

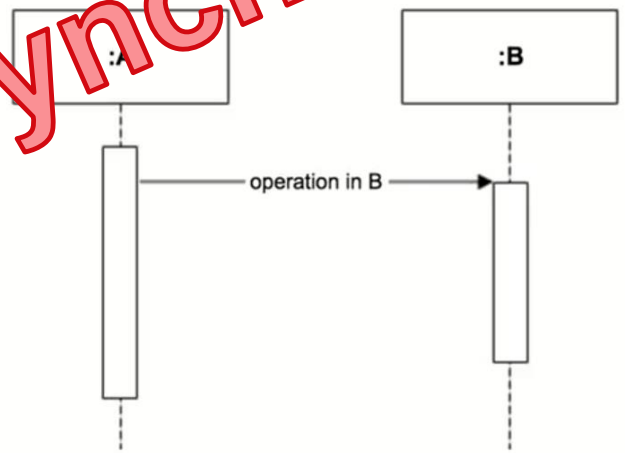


Operations and messages

Operation

When an object “calls” a method/function/operation in another object, and waits for the operation to be completed, it is represented with a filled arrowhead.

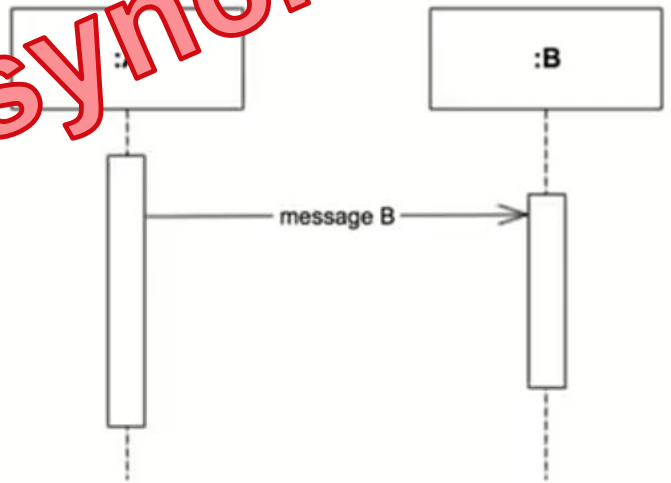
Synchronous



Message

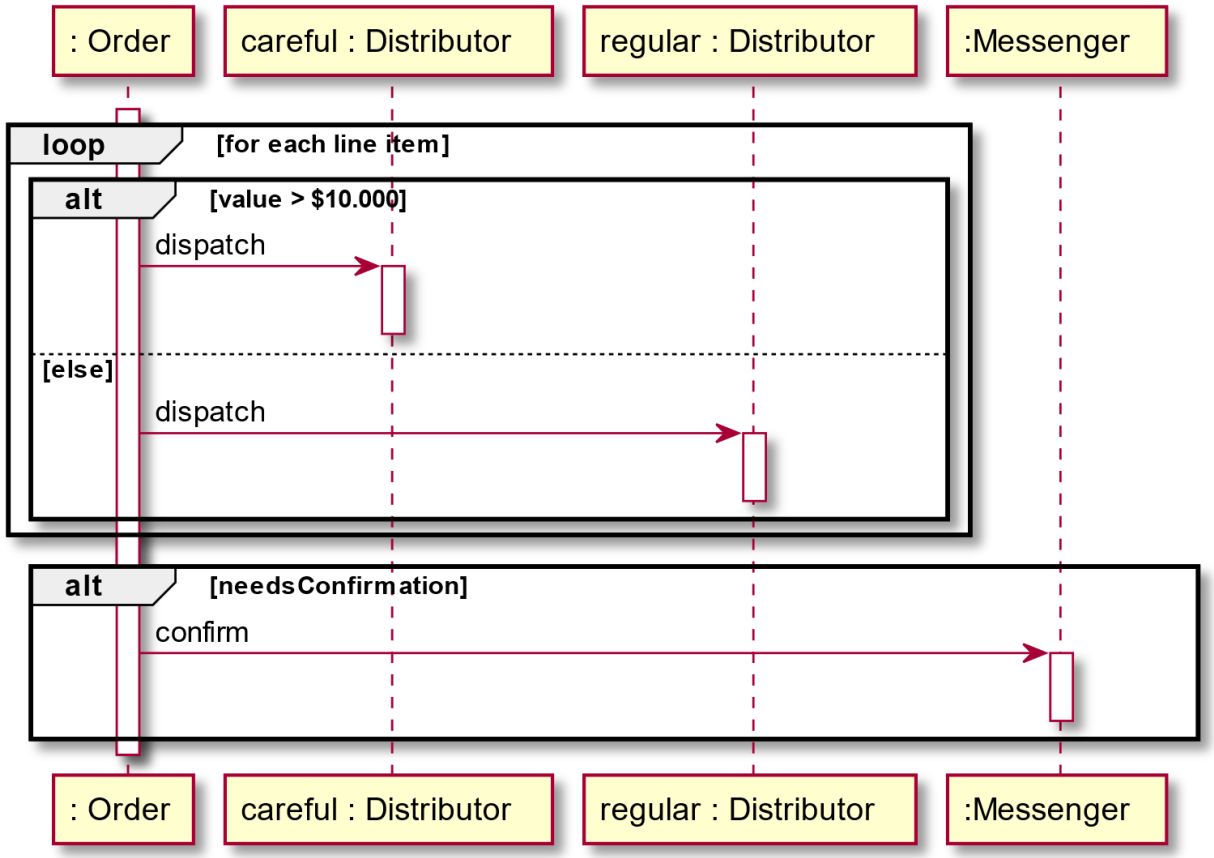
When an object “calls” a method/function/operation in another object, and doesn’t wait for the operation to be completed, it is represented with an open arrowhead.

Asynchronous



Interaction frames

- Sequence diagrams provide us with so called interaction frames.
- These interaction frames can be used to shown *alternative sequences* in the diagram, *optional sequences*, *parallel sequences*, *repeated sequences (loops)*, and *referred sequences*.



Wrapping up

- We didn't cover everything here today.
- The goal isn’t to becomeexperts in UML, but to be *literate* when it comes to the most used Diagrams, and to **know how** and **when to use them**.
- Annotations and the correct usage of symbols matters.


HÁSKÓLINN Í REYKJAVÍK
REYKJAVÍK UNIVERSITY