

OBJECT-ORIENTED PROGRAMMING

PAST, PRESENT, FUTURE

Ole Lehrmann Madsen

Aarhus University

DAHL & NYGAARD: ACM TURING AWARD WINNERS



Dahl & Nygaard at the time of Simula's development

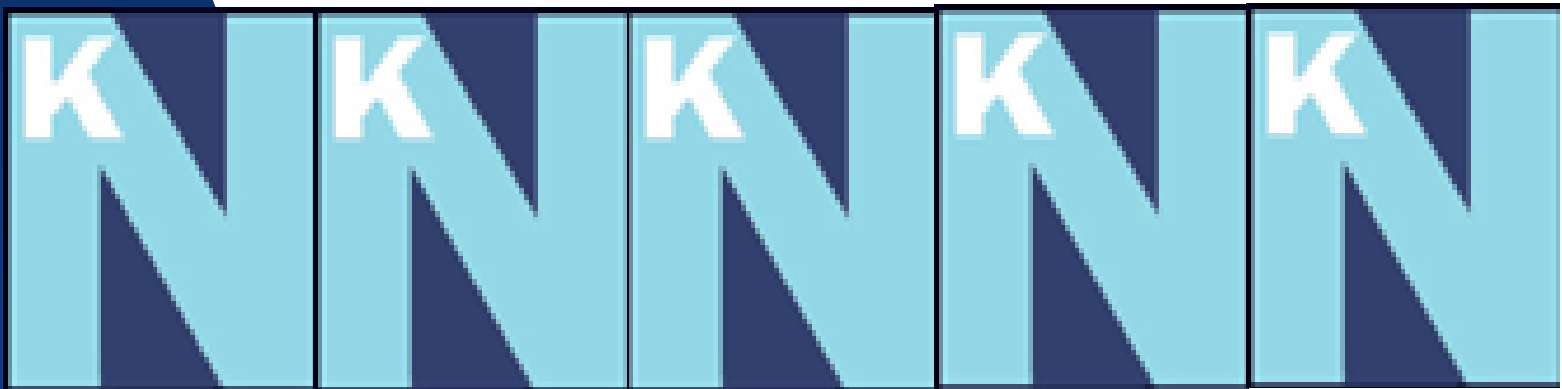
“for their role in the invention of object-oriented programming, the most widely used programming model today.” (ACM press release, 2002)



Kristen Nygaard 1926-2002

OBJECT ORIENTATION PAST, PRESENT, FUTURE

lectures by Kristen Nygaard
University of Oslo

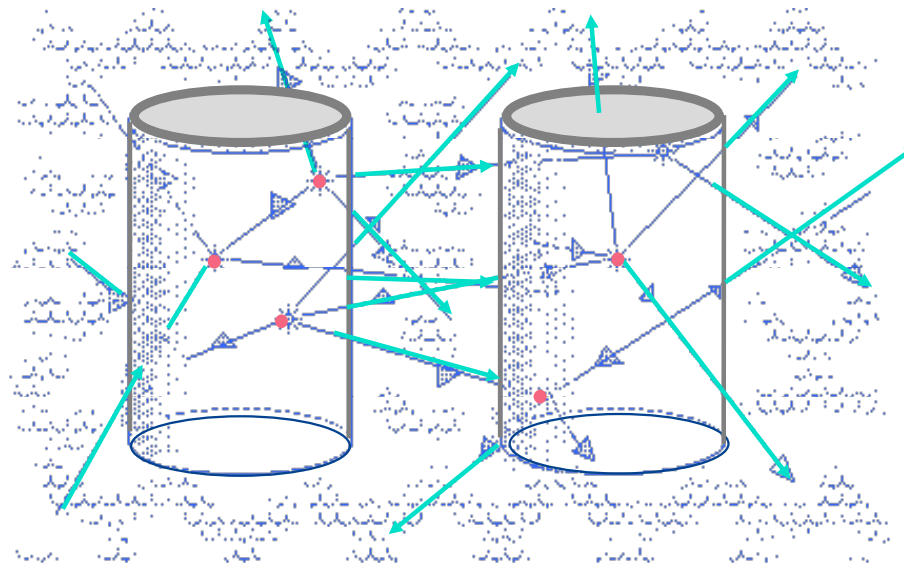




The past



THE ATOM: FROM MATHEMATICS TO MONTE CARLO





$$f(x) = g(x) + \lambda \int_{\Omega} K(x, y) f(y) dy$$



THE IDEA THAT LED TO SIMULA AND OOP:

To create a language that made it possible for people

-  to comprehend, describe, and communicate about systems
-  to analyse existing and proposed systems through computer based models



The SIMULA I language report from 1965
opens with these sentences:

“The two main objects of the SIMULA language are:

**To provide a language for a precise and standardised
description of a wide class of phenomena,
belonging to what we may call “discrete event
systems”.**

**To provide a programming language for an easy
generation of simulation programs for “discrete
event systems”.**

SIMULA I CONCEPTS

- Processes
 - corresponding to object
- Activities
 - corresponding to class





SIMULA I

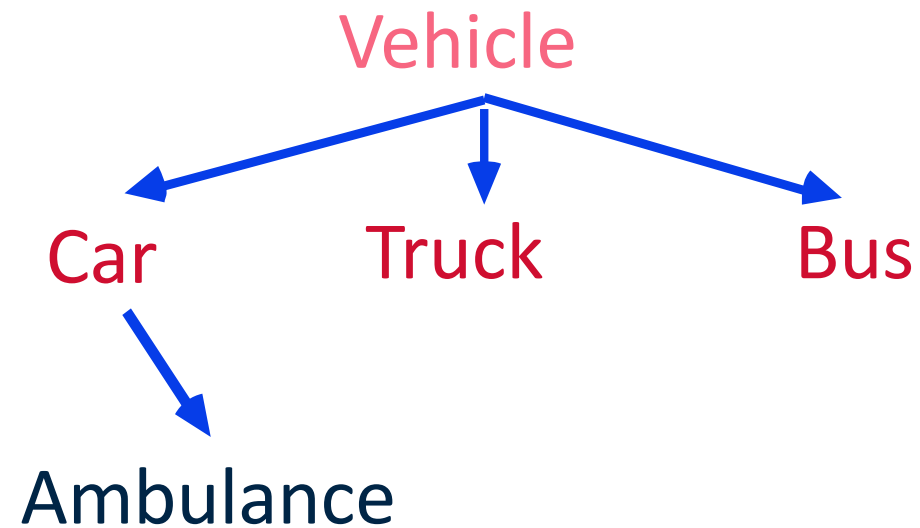
was a simulation programming language
that turned out to become
a powerful general programming language.

SIMULA 67

is a general programming language
that also is a powerful platform
for other, specialised programming languages,
as e.g. simulation languages.



SIMULA 67 was triggered off by
the invention of inheritance in January 1967.



Virtual properties

IT ALL STARTED

- with SIMULA as a language for **programming and modeling**;
- inspired by Hoare's paper on Record Handling, with **emphasis on modeling**

1. 1 Objects and Records.

... When we wish to solve a problem on a computer, we often need to construct within the computer - a model of that aspect of the real or conceptual world to which the solution of the problem will be applied. In such a model, each object of interest must be represented by some computer quantity which is accessible to the program, and which can be manipulated by it. ...

Tony Hoare: Record Handling, ALGOL Bulletin, (21), 1965

MAIN MESSAGE OF KRISTEN NYGAARD

To Program is to Understand



THE FIRST DUTY OF A PROGRAMMER

"The first duty of a revolutionary
is to get away with it."

Abbie Hoffmann

"The first duty of a programmer
is to get away with it."

Hackers' Credo

THE CONTRIBUTIONS OF SIMULA .

- Object
- Class
- Subclass
 - Single inheritance
- Virtual quantity
 - Virtual procedure (method)
- Nested/inner classes/procedures
- Action combination
 - Inner

THE CONTRIBUTIONS OF SIMULA ..

- Active objects
 - Quasi-parallel systems
 - Coroutines
 - Pre-emptive coroutines added by the *Lund SIMULA System*
- Processes and schedulers
 - Used for defining process scheduling in class Simulation
 - Later extended to concurrency
 - Necessary to be able to control scheduling
- Concurrency abstractions

THE CONTRIBUTIONS OF SIMULA ...

- The first examples of application frameworks
 - Class Simulation
- Automatic memory management
 - Garbage collection

INFLUENCE ON SIMULA

- Algol 60: a subset of Simula
- Hoare: Record Handling
- Lisp: memory management
- Conway: coroutines

INFLUENCE OF SIMULA

- Abstract data types
- Hoare: Notes on the representation of data types
- Hidden/protected added to Simula
 - Proposed by Jacob Palme
- Class invariants: Eiffel
- Algebraic data types
- Monitor (Hoare & Brinch Hansen)
- Concurrent Pascal (Brinch Hansen)
 - Class, process, monitor

OBJECT-ORIENTED LANGUAGE INFLUENCE – THE EARLY ONES

- Smalltalk
- Objective-C
- C++
- Eiffel
- Common Loops, MIT Flavors, CLOS

SMALLTALK

- Alan Kay got a tape with a Simula compiler
- More than a programming language
 - A programming system and a language
- All elements are objects
- A class is an instance of a metaclass
- All methods are virtual – can always be redefined
- No block-structure
- No coroutines, no parallel objects
- Created a large enthusiastic community

EIFFEL

- Eiffel
 - Bertrand Meyer
 - Chairman of the Simula Users Group
 - Multiple inheritance
 - Some success, but not overwhelming



C++

- Bjarne Stroustrup, student from Aarhus – datalogi 1969-1975
- Added SIMULA class, and virtual mechanisms to C
- No block-structure, no coroutines, no parallel objects
- Added multiple inheritance
- Exceptions
- Later templates and much more
- Evolved over many years
- The language that brought OO into mainstream



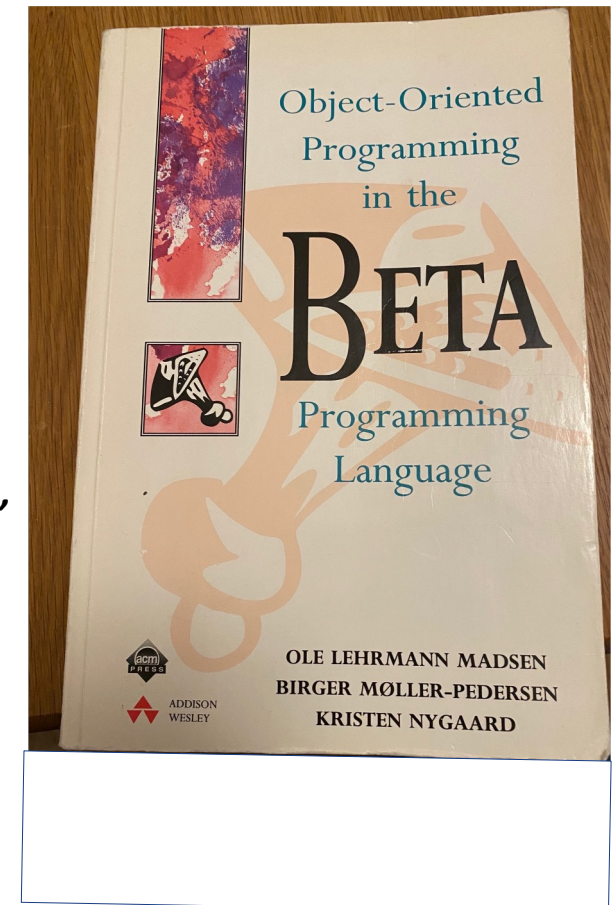
OBJECTIVE-C

- Brad Cox
- Added Smalltalk classes, and methods to C
- Many users at that time
- Used by Steve Jobs company NeXT Computer
- Used by Apple for MacOs

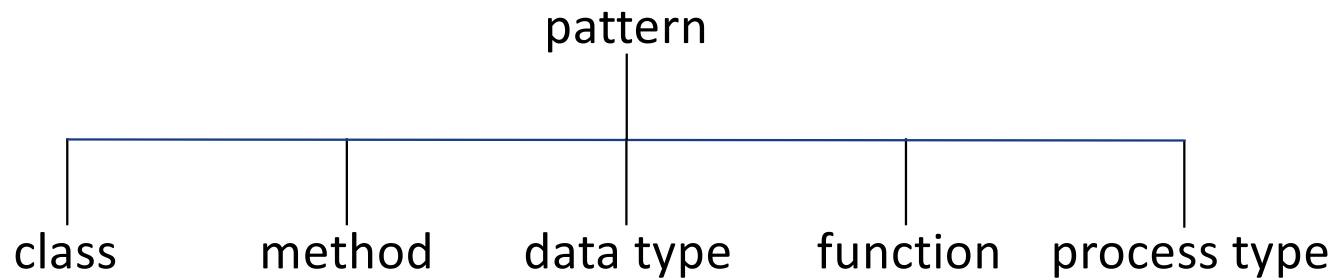
BETA

Development of the the Beta programming language

- With Bent Bruun Kristensen, Birger Møller Pedersen & Kristen Nygaard
- Beta was intended for modeling as well as programming
- Ole Lehrmann Madsen, Birger Møller-Pedersen, Kristen Nygaard.
Object-Oriented Programming in the Beta Programming Language.
ACM Press & Addison Wesley, 1993.
- Bent Bruun Kristensen, Birger Møller Pedersen, Ole Lehrmann Madsen. ***The when, why and why not of the BETA programming language.***
HOPL III: Proceedings of the third ACM SIGPLAN conference on
History of programming languages, June 2007



UNIFORM ABSTRACTION MECHANISM



- The *Pattern*
- Representing a concept
- The ultimate abstraction mechanism
- A generalization/unification of class, method, data type, function, process type, ...

BENEFITS OF THE UNIFICATION

	class	method	function	data type	process class
Pattern	+	+	+	+	+
Subpattern	+	+	+	+	+
Virtual pattern	+	+	+	+	+
Pattern variable	+	+	+	+	+
Nested pattern	+	+	+	+	+

PROTOTYPE-BASED OO LANGUAGES

- Only objects – no classes
- New objects are created by copying other objects
- Delegation is used instead of subclasses and virtuals
- ***Self***, Dave Ungar, Randy Smith, Sun Micro Systems

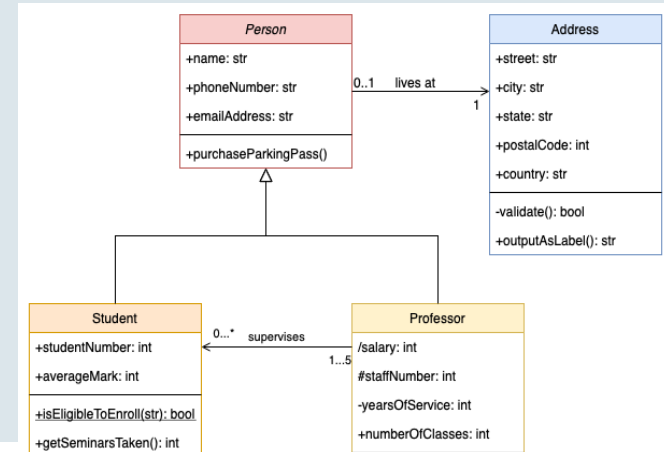
JAVA

- James Gosling, et al, Sun Micro Systems Laboratory, 1994-95
- Designed to a safe, platform independent OO programming language
- Interfaces as a simple version of multiple inheritance
- Some form of block-structure, but many restrictions
- No coroutines, no parallel objects
- Threads, but not as objects
- Half cooked version of monitors – synchronized methods

OBJECT-ORIENTED MODELING EARLY 1990'IES

- Before
 - Structured Analysis (SA), Structured Design (SD), coding
- Three different languages
- OO modeling: use of object, class, subclass, etc.
- One language for analysis, design and programming
- Use of graphical language

- Peter Coad & *Edward Yourdon*
- James Rumbaugh
- Grady Booch
- Ivar Jacobsen
- ...
- UML



RECENT OO LANGUAGES

- Python
- CLOS
- JavaScript
- PHP, *Rasmus Lerdorf*
- Lua
- Ruby on Rails, *David Heinemeier Hansson*

- C# - *Anders Hejlsberg, Mads Torgersen*
- Dart, *Lars Bak*, m.fl.
- Kotlin
- Julia
- TypeScript



HISTORY OF PROGRAMMING LANGUAGES (CONFERENCE)

HOPL I, 1978

- Fortran I, II, III, Algol, Algol 60, Lisp, COBOL, APT, Jovial, GPSS, **Simula**, JOSS, Basic, PL/1, SNOBOL, APL,

HOPL II, 1993

- Concurrent Pascal, Icon, **Smalltalk**, Algol 68, CLU, Simulation languages, Forth, C, FORMAC, Lisp, **C++**, Ada, Pascal,

HOPL III, 2007

- Modula-2, Oberon, **C++**, **Self**, **Emerald**, **Beta**, Haskell, AppleScript, **Lua**, Erlang, StateCharts, ZPL, High Performance Fortran

HOPL IV, 2020

- Clojure, MATLAB, **Groovy**, Oz, APL, Emacs Lisp, SPMD, Logo, Hygenic, **JavaScript**, LabVIEW, **D**, S&R, F#, **Smalltalk**, ML, **Objective-C**, **C++**, HDL

PRESENT THE SITUATION TODAY

CELEBRATION OF THE OO SUCCESS

- OO has been around for more than 50 years
- The dominant style of programming; most mainstream languages support it
- A substantial number of applications have been made with object-oriented programming

Object-Oriented Programming is a revolution, that, largely, we've won.

Henry Lieberman,
Panel at the 20th Anniversary of ECOOP 2006
– The Future of Object.

BLOGPOSTS

Object-Oriented Programming is The Biggest Mistake of Computer Science

Object-Oriented Programming — The Trillion Dollar Disaster

Why it's time to move on from OOP

Why is OOP Such a Waste?

Object-Oriented Programming is considered by many to be the gold standard. Yet, ironically, it is the major source of resource waste — a waste of time and money.

LOOKING INTO IT

Many posts on:

<https://betterprogramming.pub>



Ilya Suzdalnitski

Jul 10, 2019 · 27 min read · ✨

Googling OOP, you may find a lot of similar discussions, but few as radical as Suzdalnitski

Many blogposts are:

- Flawed
- Not grounded in solid knowledge
- Contain mistakes/errors

Starting to argue at this level seemed to be an endless endeavor

... AND MORE PROFOUND CRITIQUE

- The dominant object-oriented languages do not capture the intentions of its founding fathers
 - *Ole-Johan Dahl & Kristen Nygaard (SIMULA)*
'Objects (with behavior) to model complex systems, including classification in terms of classes and subclasses'
 - *Alan Kay (Smalltalk)*
'Objects being small machines communicating by message passing, no classification'

FUNCTIONAL PROGRAMMING

- More and more focus on functional programming
- Templates for defining generic classes and methods
- Functional interfaces and lambdas in Java
- Scala, support for OO and FP
- FP is not necessarily an alternative
- ***OO and FP can easily supplement each other***
- Nygaard: one did not drop *addition* when *multiplication* was invented;-)
- Beta did have support for some form of functional programming:
 - Higher order types (patterns), pattern as first-class values, etc.
 - But not as elegant as many modern FPs.

THE SITUATION TODAY

- Too much focus on code reuse
- Too little focus on modeling

- Inheritance: a technique for reuse
- Many problematic issues with mainstream OOP

- The programming communities and modeling communities are almost disjoint

- We advocate *going back to the future*:
 - Re-introduce modeling as the main benefit of object-oriented programming
 - Reuse is a nice side-effect, but secondary

PROBLEMATIC ISSUES IN MAINSTREAM OO

- The programming communities and modeling communities are almost disjoint
- **Code reuse** is seen as the main benefit of OO
 - Inheritance is seen as a technique for reuse of code in the superclass
 - Implementation inheritance
 - Smalltalk, Java: all methods are virtual
 - ...
- **Modeling** is rarely mentioned by most books on OO
 - No teaching of modeling
 - Separate community om modeling

QUOTES FROM SOME PROMINENT PEOPLE

A highly influential paper:

*Inheritance as an incremental
modification technique*

Wegner & Zdonik, ECOOP 1989

*The philosophical viewpoint that “objects
model the real world” has never
appealed to me.*

William Cook: Peak Objects, Special 20th
Anniversary Session at ECOOP 2006

*The ultimate goal of object-oriented
programming should be code reuse.*

Bono et al, ECOOP 2012

*Inheritance has stood the test of time as a useful feature,
and I now believe that some code-reuse mechanism like
inheritance or delegation is a very important part of any
object-oriented programming system.*

Andrew Black: Object-oriented programming:
Some history, and challenges for the next fifty
years, Information and Computation, 2013

MODELING IS NOT EXPLICIT IN MAINSTREAM OOP

- Most textbooks on OOP are quite technical
- Modeling is often implicit via examples

If you write a computer program in an object-oriented language, you are creating, in your computer, a model of some part of the world. ...

D.J. Barnes, M. Kölling: Objects First with Java – A practical introduction using BlueJ, sixth edition, Pearson, 2017

IT ALL STARTED

- with SIMULA as a language for **programming and modeling**;
- inspired by Hoare's paper on Record Handling, with **emphasis on modeling**

1. 1 Objects and Records.

... When we wish to solve a problem on a computer, we often need to construct within the computer - a model of that aspect of the real or conceptual world to which the solution of the problem will be applied. In such a model, each object of interest must be represented by some computer quantity which is accessible to the program, and which can be manipulated by it. ...

Tony Hoare: Record Handling, ALGOL Bulletin, (21), 1965

AND MODELING AS PART OF OOP HAS BEEN THERE SINCE

With benefits, e.g. that programs reflect application domains

The basic philosophy underlying object-oriented programming is to make the programs as far as possible reflect that part of the reality they are going to treat. It is then often easier to understand and to get an overview of what is described in programs.

Krogdahl and Olsen, 1986

Object-oriented analysis is based upon concepts that we first learned in kindergarten: objects and attributes, classes and members, wholes and parts.

Coad and Yourdon, Object-Oriented Analysis,
1990

By focusing on reuse, the benefit of modeling is lost



MAIN MESSAGE OF KRISTEN NYGAARD

To Program is to Understand

■ WHAT DO WE UNDERSTAND BY MODELING?

PROGRAMMING AS MODELING

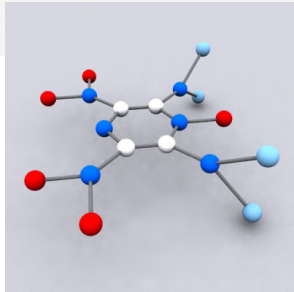
- For Beta we developed a ***conceptual framework*** for organizing and understanding knowledge about application domains
- This includes:
 - A definition of ***object-oriented model***
 - Understanding ***phenomena*** and ***concepts***
 - Understanding conceptual means such as ***classification, composition, context, association***

MODELLING IN GENERAL

Webster:

A model refers to a small or large, abstract or actual, representation of a planned or existing entity or system from a particular viewpoint.

- **Abstract:**
 - A mathematical description
 - ***Newton's laws of motion***
- **Actual:**
 - Something physical
 - A molecule:



Model of planned building



Model of (existing) solar system

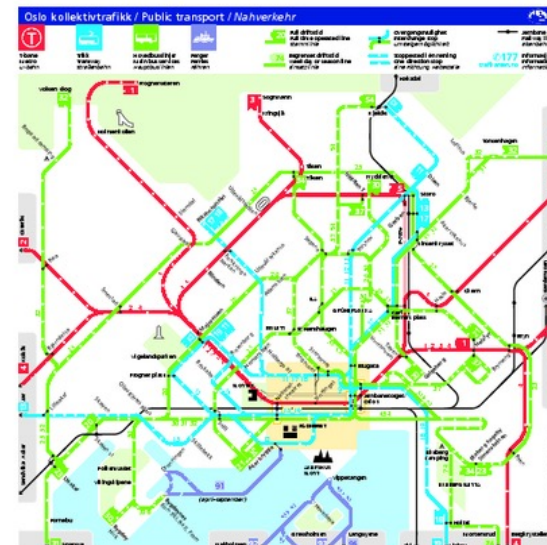


FROM A PARTICULAR VIEWPOINT

A good map provides the information we need for a particular purpose – or the information the mapmaker wants us to have

Peter Turchi:

Maps of the Imagination – the writer as a cartographer



THE PROGRAM EXECUTION IS THE MODEL

Object-oriented programming: A program execution is regarded as a **physical model**, representing the behavior of either an **existing** or **planned** system (in an application domain) from a particular **viewpoint**.

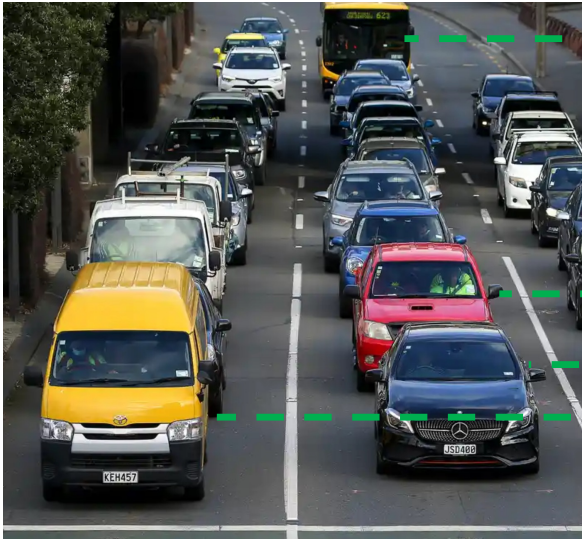
The program execution:
The dynamic system of objects generated during execution of the program

The program execution is the model

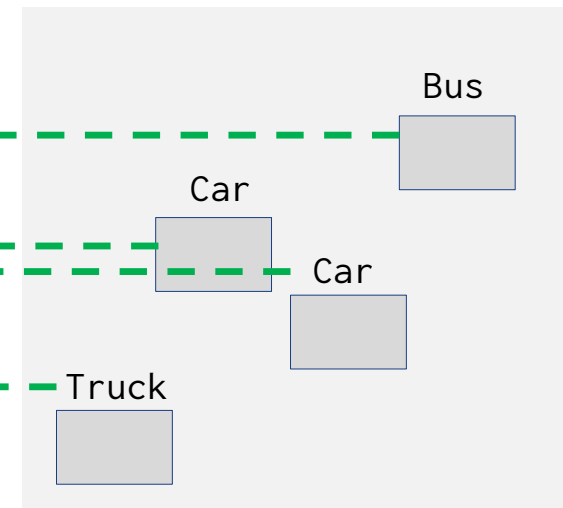
description/program text

```
class Vehicle: ...  
class Car: Vehicle ...  
class Bus: Vehicle ...  
class Truck: Vehicle ...  
class Ambulance: Car ...
```

application domain



program execution with objects



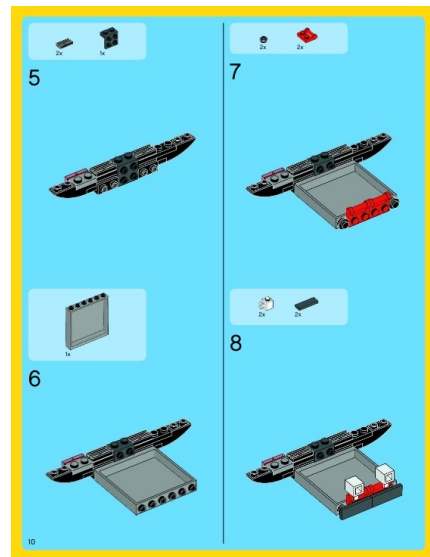
represents

model of

Which of these is a model of the Mini?



Domain

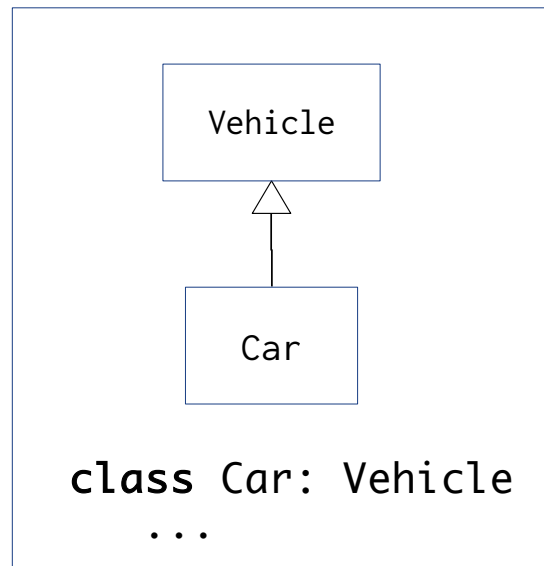


or

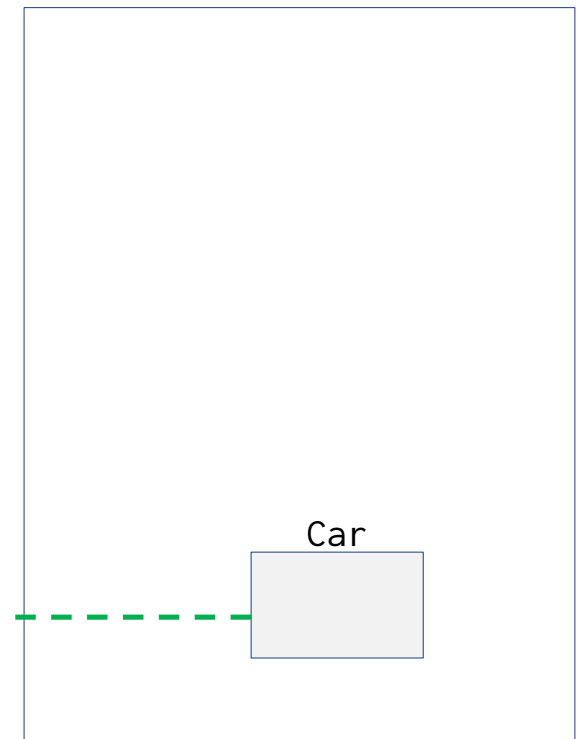


Model

The Car object is a model of the Mini



Description



Model

PHENOMENA

application domain



A *phenomenon* is

- a thing that exists in reality or in the mind

Kinds of phenomena

- Each single car is a *physical* phenomenon
- The weight, and license number of a car are *properties* of a of physical phenomenon
- The car trip from home to work is a *transformation* that changes the state (properties) of a physical object (a car)

CONCEPT

application domain



A **concept** is

- a generalized idea of a collection of phenomena, based on knowledge of common properties of instances in the collection

Examples domain concepts

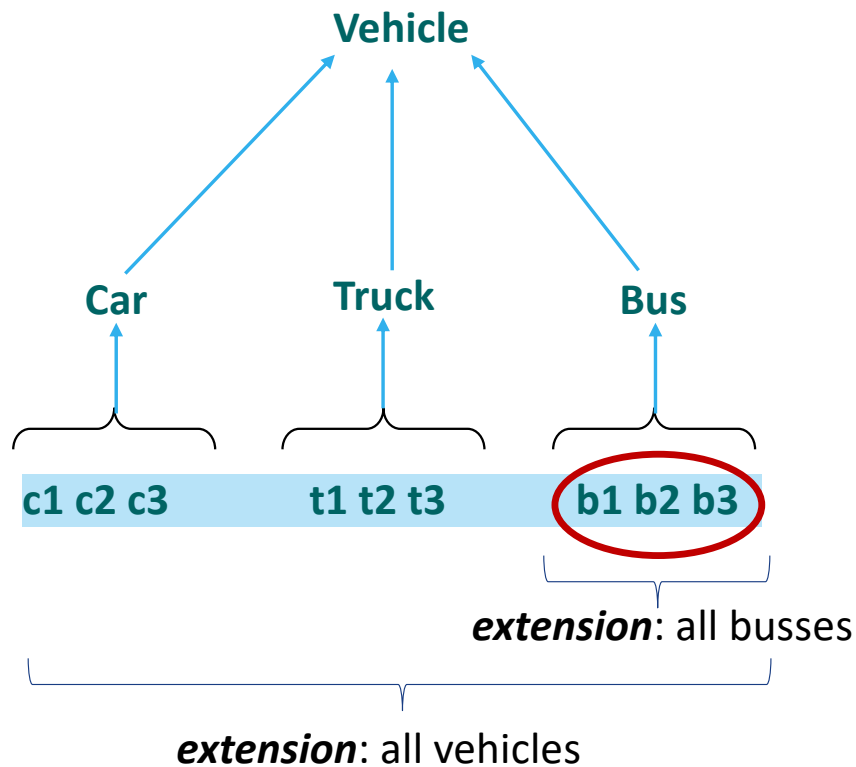
- Vehicle, car, bus, truck, car, ambulance, ...
- KerbWeight, ...
- Trip, ...
- VehicleRegister, Registration, ...

CONCEPTUAL MEANS

Conceptual means for understanding and organizing knowledge

- ***Classification***
 - Organizing phenomena and concepts in hierarchies
- ***Composition***
 - Understanding the structure of physical objects as parts and wholes
- ***Context***
 - Understanding the context of phenomena and concepts
- ***Association***
 - Relations between phenomena

CONCEPTS AND CLASSIFICATION



- A **concept** is a generalized idea of a collection of phenomena
 - The **extension** of concept is the phenomena covered by the concept
 - The **intension** of a concept is the properties of phenomena covered by the concept
 - Concepts may be organized in **generalization/specialization** hierarchies
-
- The extension of a specialized concept is a **subset** of the extension of the corresponding generalized concept
 - The intension of a specialized concept **extends** the intension of the corresponding generalized concept by adding additional properties

REPRESENTATION OF CONCEPTS

```
class Vehicle
    // attributes representing
    // the intension of vehicles
```

```
class Bus: Vehicle
    // the attributes (intension)
    // of Vehicle are inherited
    // and additional attributes
    // of busses are added
```

- Classes may represent concepts
- Subclasses may represent specializations of a given concept
- A class hierarchies may represent a classification hierarchies
- Objects of a class may represent phenomena

The *essence*:

- properties specified in a class are inherited/imposed on subclasses

MULTIPLE INHERITANCE AND CLASSIFICATION

Issue

- Multiple inheritance, mixins, and traits add complexity and ambiguity compared to single inheritance.

when used for code reuse

Is a given case an alternative classification hierarchy?

Properties of classification hierarchies:

- Tree-structured classification hierarchy
- Non-tree structured classification hierarchy
- Several independent classification hierarchies
- Dynamic classification
- Classification of actions
 - Inheritance for methods and processes

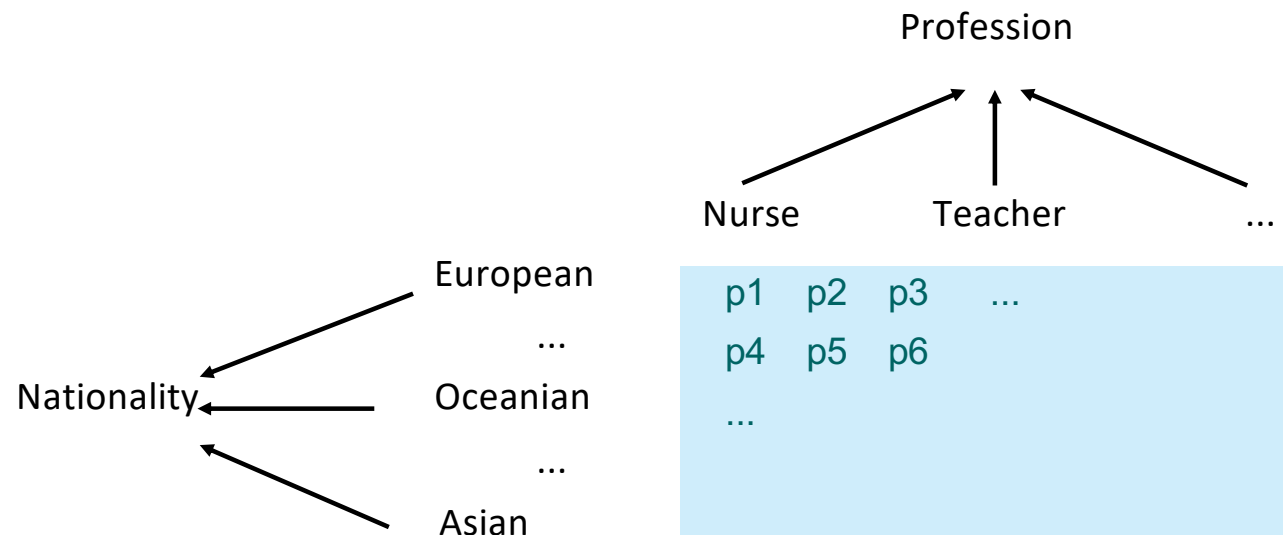
Madsen & Møller-Pedersen, ECOOP87

See Madsen, Møller-Pedersen, Nygaard: *Object-oriented programming in the Beta programming language*, Addison

Wesley, ACM Press, 1993

SEVERAL CLASSIFICATION HIERARCHIES

Independent tree-structured classifications of the same objects



WHAT ABOUT OO MODELING?

- OO modeling (Coad & Yourdon, Rumbaugh et al, Booch, ..., UML) replaced Structured Analysis and Structured Design
- Same approach throughout all development phases
- The bad news:
 - Modeling and programmings communities became two disjoint communities
 - Programmers don't make UML models
 - Modelers don't make programs?
 - Many projects start with a model, but often left behind, since no time to update
 - models become obsolete, code counts

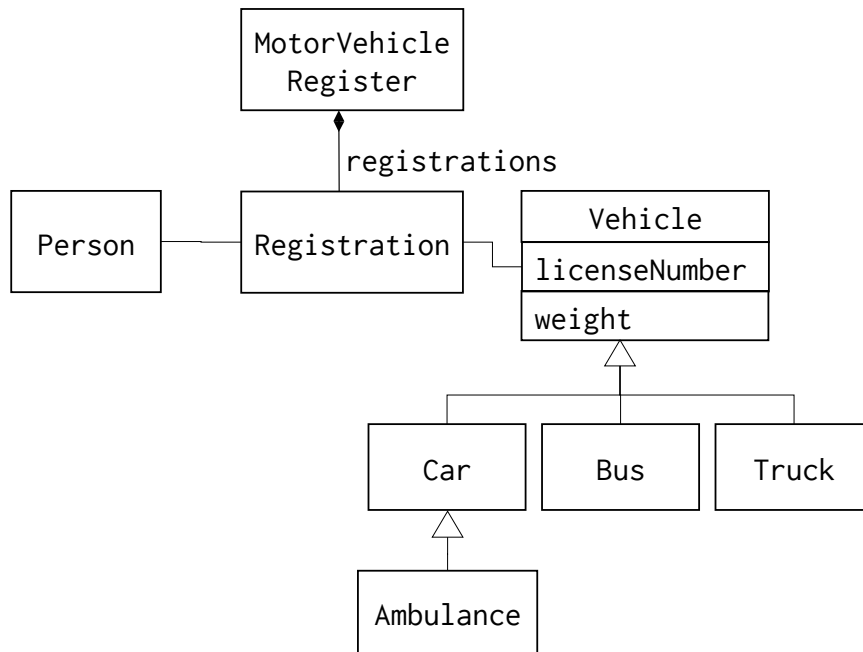
THE GOOD NEWS

You do not have to use UML for real for modeling

You may use your favorite OO language to describe the model

TEXT VERSUS DIAGRAMS

The textual description
has as much information as the diagrams



```
class MotorVehicleRegister:
    registrations:...
```

```
...
class Registration:
    ...
```

```
class Vehicle:
    licenseNumber:...
    weight:...
    ...
```

```
class Person:
    ...
```

```
class Car: Vehicle
```

```
...
```

```
class Bus: Vehicle
```

```
...
```

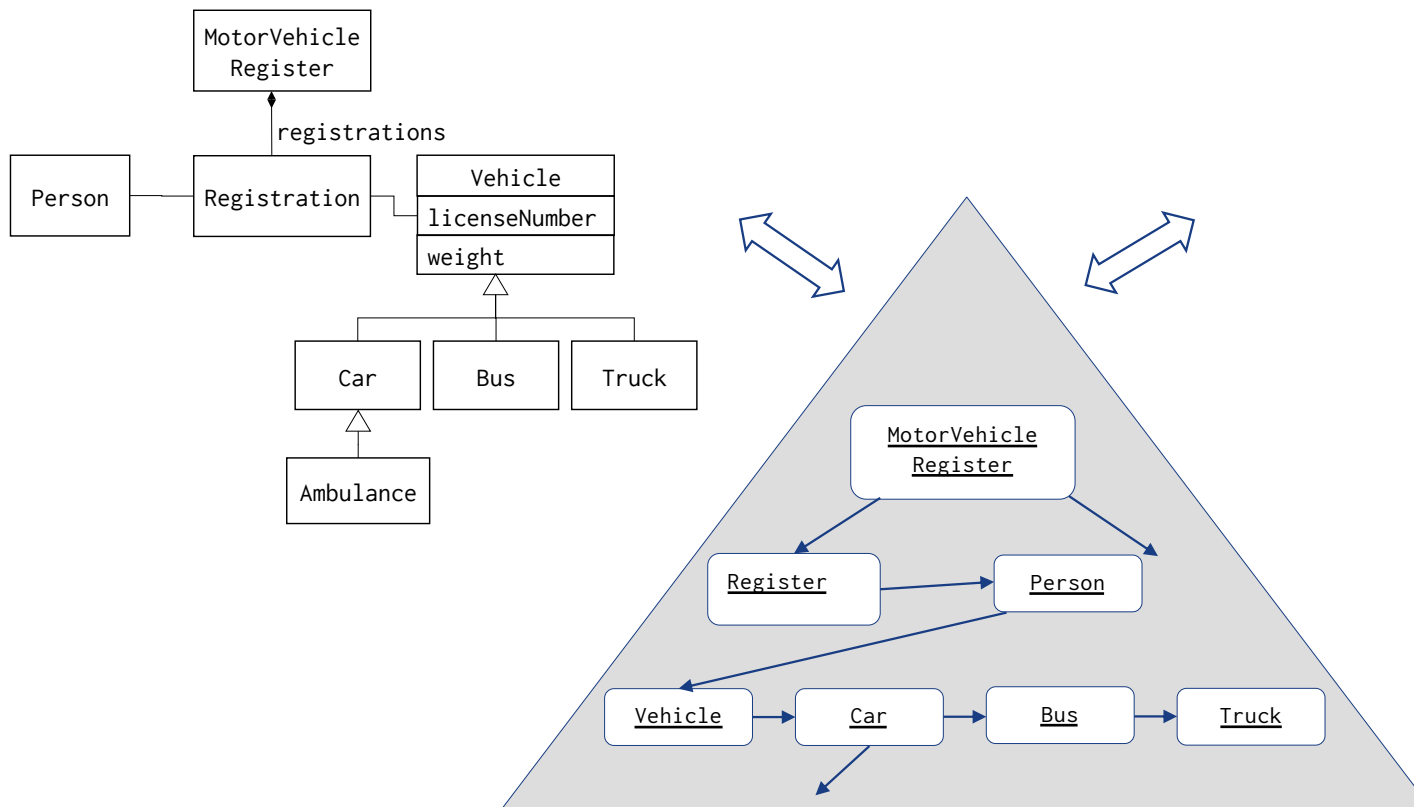
```
class Truck: Vehicle
```

```
...
```

```
class Ambulance: Car
```

```
...
```


MJØLNER BETA TOOL: TEXT VIEW AND DIAGRAM VIEW



```
class MotorVehicleRegister:
    registrations:...
    ...
class Registration:
    ...
class Vehicle:
    licenseNumber:...
    weight:...
    ...
class Person: ...
class Car: Vehicle ...
class Bus: Vehicle ...
class Truck: Vehicle ...
class Ambulance: Car ...
```

Abstract Syntax Tree:
Common Representation

THE FUTURE?

- More focus on OO modeling
 - Use the same language for modeling and programming
- Better support for parallel programming
 - Active objects as in SIMULA and Beta
 - Support for building safe parallel abstractions
- Better integration of functional programming
 - For state-transitions in objects
 - Scala, Beta,
- Better support for classification, composition, context, association
- Better support for generics
 - No template mechanism as a compile-time programming languages

BACK TO THE FUTURE: REINTRODUCE MODELING AS THE MAIN BENEFIT OF OO

- *What Object- Oriented Programming Was Supposed to Be: Two Grumpy Old Guys' Take on Object-Oriented Programming*,
Ole Lehrmann Madsen, Birger Møller-Pedersen,
Onward! 2022, part of OOPSLA/SPLASH 2022. Auckland, New Zealand
- *What your mother forgot to tell you about modeling – and programming*,
Ole Lehrmann Madsen, Birger Møller-Pedersen,
2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems
- *An Introduction to Object-Oriented Programming as Modeling*
 - Book in progress
 - <https://oopm.org>
- Development of a new language **qBeta**