Object-Oriented Programming in MATLAB

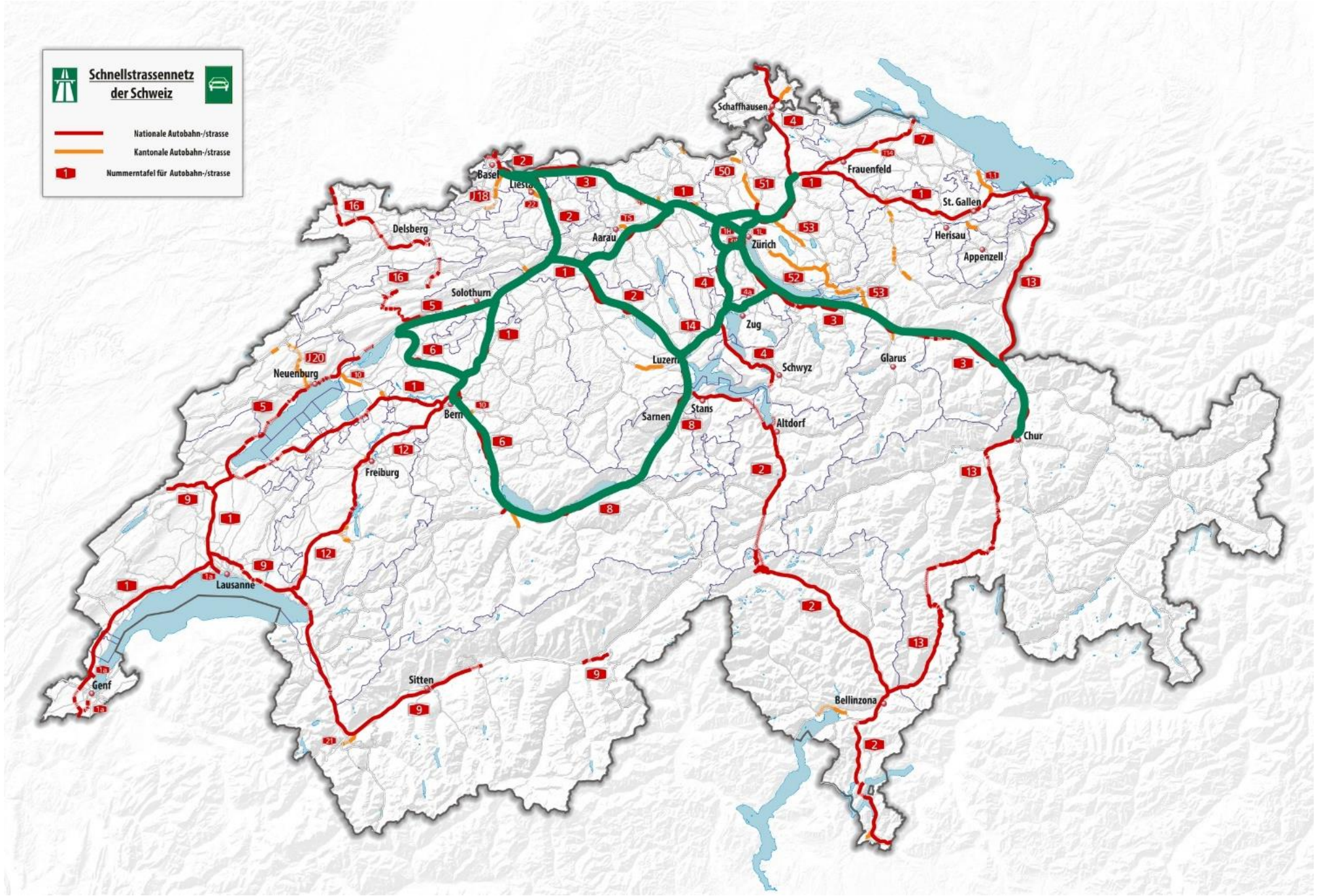
Benjamin Müller

Task

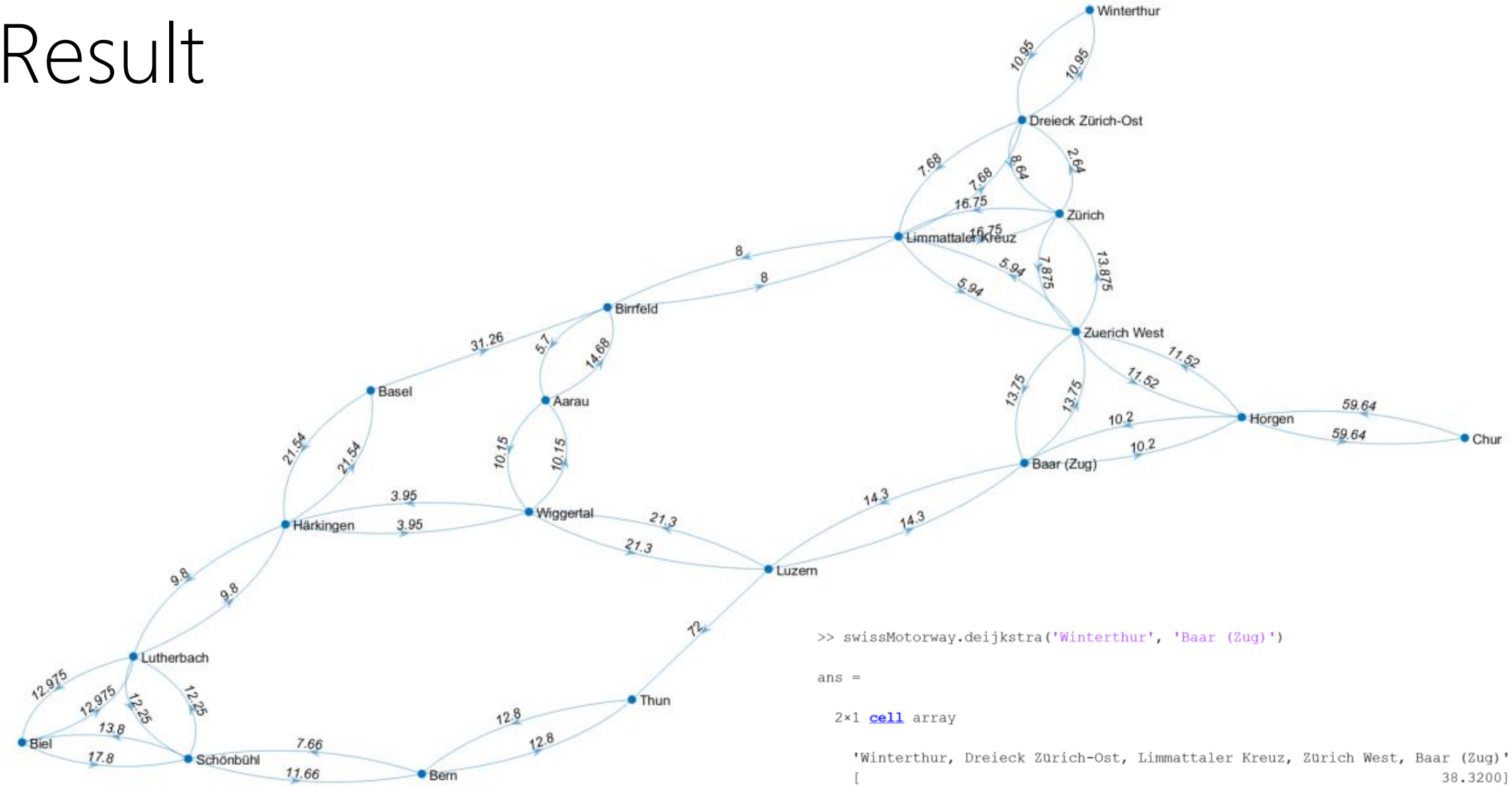
Ein Objektmodell für das Schweizerische Autobahnnetz erstellen und mit realitätsnahen Daten initialisieren. Anhand dieses Autobahnnetztes (Graphs), zwei verschiedene Algorithmen implementieren welche die folgenden Probleme lösen:

- Minimal spanning tree
  Die optimale Route zum Besuchen aller Städte finden.
- Shortest path
  Die optimale Route zwischen zwei Städten finden.

Task


Schnellstrassennetz der Schweiz
Nationale Autobahn-/strasse
Kantonale Autobahn-/strasse
Nummerntafel für Autobahn-/strasse
Schaffhausen
Frauenfeld
St. Gallen
Herisau
Appenzell
Basel
Liestal
Delsberg
Aarau
Zürich
Solothurn
Zug
Luzern
Schwyz
Glarus
Neuenburg
Bern
Sarnen
Stans
Altdorf
Chur
Freiburg
Lausanne
Genf
Sitten
Bellinzona

Result



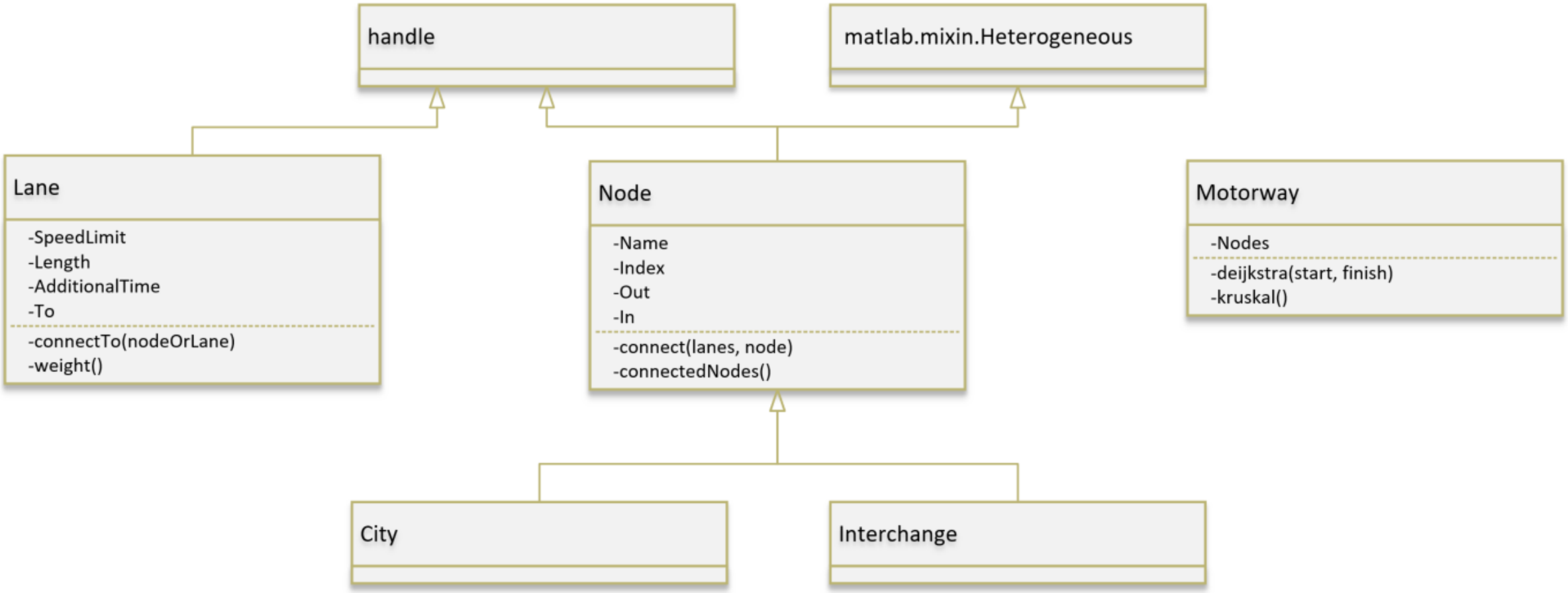

```
>> swissMotorway.deijkstra('Winterthur', 'Baar (Zug)')

ans =

  2×1 cell array

    'Winterthur, Dreieck Zürich-Ost, Limmattaler Kreuz, Zürich West, Baar (Zug)'
    [                                                                    38.3200]
```

Result


handle
matlab.mixin.Heterogeneous
Lane
-SpeedLimit
-Length
-AdditionalTime
-To
-connectTo(nodeOrLane)
-weight()
Node
-Name
-Index
-Out
-In
-connect(lanes, node)
-connectedNodes()
Motorway
-Nodes
-deijkstra(start, finish)
-kruskal()
City
Interchange

Hürden MATLAB OOP

- «List comprehension functions» ungeeignet für Objekte

```
lanes = [
    Model.Lane(100, 1, 10)
    Model.Lane(80, 1, 0)
    Model.Lane(120, 1, 5)
];

nums = [
    5
    1
    -10
];

% Map
speedLimits = arrayfun(@(x) x.SpeedLimit, lanes);

% Sort
sorted = sort([5 1 -10 8 9 4 10]);
sortedLanes = sort(lanes, @(x) x.SpeedLimit); % ???

% Filter/Where
filtered = nums(nums > 0);
filteredLanes = lanes(lanes.AdditionalTime == 0); % ???
```

Hürden OOP MATLAB

- «List comprehension functions» ungeeignet für Objekte
- Handle- und Object-Klassen kaum unterscheidbar

```
lane = Model.Lane(100, 1, 10);
x = lane;
lane.SpeedLimit = 120;
y = lane;

disp(x.SpeedLimit); % displays 100
disp(y.SpeedLimit); % displays 120


lane = Model.Lane(100, 1, 10); % Model.Lane < handle
x = lane;
lane.SpeedLimit = 120;
y = lane;

disp(x.SpeedLimit); % displays 120
disp(y.SpeedLimit); % displays 120
```

OOP MATLAB, the good parts

- Closures

```
classdef Mensch
    properties
        Name = 'Peter'
    end

    methods
        function x = definiereEssen(obj, essen)
            anz = 0;
            x = @do;

            function do()
                anz = anz + 1;
                fprintf('%s isst %s, er hat schon %d gehabt.\n', obj.Name, essen, anz);
            end
        end
    end
end
```

```
peter = Model.Mensch();
kuchen = peter.definiereEssen('Kuchen');
pizza = peter.definiereEssen('Pizza');
kuchen(); % Peter isst Kuchen, er hat schon 1 gehabt.
pizza();  % Peter isst Pizza, er hat schon 1 gehabt.
kuchen(); % Peter isst Kuchen, er hat schon 2 gehabt.
kuchen(); % Peter isst Kuchen, er hat schon 3 gehabt.
pizza();  % Peter isst Pizza, er hat schon 2 gehabt.
```

OOP MATLAB, the good parts

- Closures
- Operator overloading

```
function r = plus(obj1, obj2)
    r = horzcat(obj1, obj2);
end

schoenbuehl.connect(Model.Lane(100, 1.5, 0) + Model.Lane(120, 23.8, 1), biel);
```

Fragen / Inputs