CARL VON OSSIETZKY universität OLDENBURG

*Sommersemester 2020*

# Objektorientierte Modellierung und Programmierung

Dr. Christian Schönberg

CARL VON OSSIETZKY universität OLDENBURG

# Großübung
# Programmierparadigmen

CARL VON OSSIETZKY universität OLDENBURG

# Standard ML

CARL VON OSSIETZKY universität OLDENBURG

# Listen-Funktionen

```
fun length (nil)    = 0
  | length (x::xs) = 1 + length(xs);
```

CARL VON OSSIETZKY universität OLDENBURG

# Listen-Funktionen

```
fun length (nil)    = 0
  | length (x::xs) = 1 + length(xs);
```

```
val length = fn : 'a list -> int
```

CARL VON OSSIETZKY universität OLDENBURG

# Listen-Funktionen

```
fun length (nil)   = 0
  | length (x::xs) = 1 + length(xs);
```

```
val length = fn : 'a list -> int
```

```
fun position(x, nil)   = ~1
  | position(x, y::ys) = if   x=y
                         then 0
                         else 1 + position(x, ys);
```

CARL VON OSSIETZKY universität OLDENBURG

# Listen-Funktionen

```
fun length (nil)   = 0
  | length (x::xs) = 1 + length(xs);
```

```
val length = fn : 'a list -> int
```

```
fun position(x, nil)   = ~1
  | position(x, y::ys) = if   x=y
                         then 0
                         else 1 + position(x, ys);
```

```
val position = fn : ''a * ''a list -> int
```

# Listen-Funktionen

```
fun length (nil)   = 0
  | length (x::xs) = 1 + length(xs);
```

```
val length = fn : 'a list -> int
```

```
fun position(x, nil)   = ~1
  | position(x, y::ys) = if   x=y
                         then 0
                         else 1 + position(x, ys);
```

```
fun position(x, nil)   = ~1
  | position(x, y::ys) =
        if x=y then 0
               else if   position(x, ys) = ~1
                    then ~1
                    else position(x, ys) + 1;
```

```
val position = fn : ''a * ''a list -> int
```

CARL VON OSSIETZKY universität OLDENBURG

# Listen-Funktionen (2)

- Definieren Sie eine Funktion **`find`**, die als Parameter eine Funktion und eine Liste bekommt. Wenn die übergebene Funktion angewandt auf mindestens ein Element der Liste **`true`** zurückgibt, dann gibt **`find`** ebenfalls **`true`** zurück, sonst gibt **`find false`** zurück.

```
val find = fn : ('a -> bool) * 'a list -> bool
```

CARL VON OSSIETZKY universität OLDENBURG

# Listen-Funktionen (2)

- Definieren Sie eine Funktion **`find`**, die als Parameter eine Funktion und eine Liste bekommt. Wenn die übergebene Funktion angewandt auf mindestens ein Element der Liste **`true`** zurückgibt, dann gibt **`find`** ebenfalls **`true`** zurück, sonst gibt **`find false`** zurück.

```
fun find(f, nil)    = false
  | find(f, x::xs) = if   f(x)
                     then true
                     else find(f, xs);
```

```
val find = fn : ('a -> bool) * 'a list -> bool
```

# Aggregats-Funktionen

- Definieren Sie eine Funktion **`sum`**, welche die Summe einer Liste von **`int`**-Werten berechnet.
- Definieren Sie eine Funktion **`avg`**, die den Durchschnittswert (arithmetisches Mittel) einer Liste von **`int`**-Werten berechnet.
  - Integer-Division mit **`div`**
  - $avg = \frac{\sum_{i=0}^{n-1} list_i}{|list|}$

# Aggregats-Funktionen

- Definieren Sie eine Funktion **`sum`**, welche die Summe einer Liste von **`int`**-Werten berechnet.
- Definieren Sie eine Funktion **`avg`**, die den Durchschnittswert (arithmetisches Mittel) einer Liste von **`int`**-Werten berechnet.
  - Integer-Division mit **`div`**
  - $avg = \frac{\sum_{i=0}^{n-1} list_i}{|list|}$

```
fun sum(nil) = 0
  | sum(x::xs) = x + sum(xs);
```

```
fun avg(l) = sum(l) div length(l);
```

CARL VON OSSIETZKY universität OLDENBURG

# Exceptions

```
exception Empty;

fun nth(nil, n)    = raise Empty
  | nth(x::xs, 0) = x
  | nth(x::xs, n) = nth(xs, n-1);
```

# Aggregats-Funktionen (2)

- Definieren Sie eine Funktion **`median`**, die den Median einer *sortierten* Liste von **`int`**-Werten berechnet.
  - ungerade Anzahl von Werten: Median ist der mittlere Wert
  - gerade Anzahl von Werten: Median ist das arithmetische Mittel der beiden mittleren Zahlen

  - Negation mit **`not(x)`**
  - Modulo mit **`mod`**

# Aggregats-Funktionen (2)

- Definieren Sie eine Funktion **`median`**, die den Median einer *sortierten* Liste von **`int`**-Werten berechnet.
  - ungerade Anzahl von Werten: Median ist der mittlere Wert
  - gerade Anzahl von Werten: Median ist das arithmetische Mittel der beiden mittleren Zahlen

```
fun isOddLength(nil) = false
  | isOddLength(x::xs) = not(isOddLength(xs));
fun isOddLength(l) = length(l) mod 2 = 1;
```

CARL VON OSSIETZKY universität OLDENBURG

# Aggregats-Funktionen (2)

- Definieren Sie eine Funktion **`median`**, die den Median einer *sortierten* Liste von **`int`**-Werten berechnet.
  - ungerade Anzahl von Werten: Median ist der mittlere Wert
  - gerade Anzahl von Werten: Median ist das arithmetische Mittel der beiden mittleren Zahlen

```
fun isOddLength(nil) = false
  | isOddLength(x::xs) = not(isOddLength(xs));
fun isOddLength(l) = length(l) mod 2 = 1;

fun median(nil) = 0
  | median(l) = if isOddLength(l)
                then nth(l, length(l) div 2)
                else avg(nth(l, length(l) div 2 - 1)::
                         nth(l, length(l) div 2)::nil);
```

# Datenauswertung

Datenschema: Name, Alter, Kontostand

CARL VON OSSIETZKY universität OLDENBURG

# Datenauswertung

Datenschema: Name, Alter, Kontostand

```
(* schema: name, age, money *)
val db = [("Donald", 24, 0),
          ("Dagobert", 58, 100000000),
          ("Mickey", 31, 1000),
          ("Goofy", 32, 100)];
```

CARL VON OSSIETZKY universität OLDENBURG

# Datenauswertung

Datenschema: Name, Alter, Kontostand

```
(* schema: name, age, money *)
val db = [("Donald", 24, 0),
          ("Dagobert", 58, 100000000),
          ("Mickey", 31, 1000),
          ("Goofy", 32, 100)];
```

```
val db = [("Donald",24,0),
          ("Dagobert",58,100000000),
          ("Mickey",31,1000),
          ("Goofy",32,100)] :
          (string * int * int) list
```

CARL VON OSSIETZKY universität OLDENBURG

# Datenauswertung

Datenschema: Name, Alter, Kontostand

```
val db = [("Donald",24,0),("Dagobert",58,100000000),…]
```

```
fun sumMoney(nil) = 0
  | sumMoney((n,a,m)::xs) = m + sumMoney(xs);
```

```
val getNames = fn : ('a * 'b * 'c) list -> 'a list
```

CARL VON OSSIETZKY universität OLDENBURG

# Datenauswertung

Datenschema: Name, Alter, Kontostand

```
val db = [("Donald",24,0),("Dagobert",58,100000000),…]
```

```
fun sumMoney(nil) = 0
  | sumMoney((n,a,m)::xs) = m + sumMoney(xs);

fun getNames(nil) = []
  | getNames((n,a,m)::xs) = n::getNames(xs);
```

CARL VON OSSIETZKY universität OLDENBURG

# Datenauswertung

Datenschema: Name, Alter, Kontostand

```
val db = [("Donald",24,0),("Dagobert",58,100000000),…]
```

```
fun sumAge(nil) = 0
  | sumAge((n,a,m)::xs) = a + sumAge(xs);
fun avgAge(nil) = 0
  | avgAge(l) = sumAge(l) div length(l);

fun getDagobertsMoney(nil) = 0
  | getDagobertsMoney((n,a,m)::xs) =
        if String.compare(n,"Dagobert")=EQUAL
        then m
        else getDagobertsMoney(xs);
```

```
val change =
  fn : ('a * int * 'b) list -> ('a * int * 'b) list
(* Alter durch Geburtsjahr ersetzen: 2020 – Alter *)
```

CARL VON OSSIETZKY universität OLDENBURG

# Datenauswertung

Datenschema: Name, Alter, Kontostand

```
val db = [("Donald",24,0),("Dagobert",58,100000000),…]
```

```
fun sumAge(nil) = 0
  | sumAge((n,a,m)::xs) = a + sumAge(xs);
fun avgAge(nil) = 0
  | avgAge(l) = sumAge(l) div length(l);

fun getDagobertsMoney(nil) = 0
  | getDagobertsMoney((n,a,m)::xs) =
        if String.compare(n,"Dagobert")=EQUAL
        then m
        else getDagobertsMoney(xs);

fun change(nil) = []
  | change((n,a,m)::xs) = (n,2020-a,m)::change(xs);
```

CARL VON OSSIETZKY universität OLDENBURG

# Prolog

CARL VON OSSIETZKY universität OLDENBURG

# Relationen

```
append([], L, L).
append([J|K], L, [J|KL]) :- append(K, L, KL).
```

CARL VON OSSIETZKY universität OLDENBURG

# Relationen

```
append([], L, L).
append([J|K], L, [J|KL]) :- append(K, L, KL).
```

```
K = [2,3]
L = [4,5]
KL = [2,3,4,5]
```

CARL VON OSSIETZKY universität OLDENBURG

# Relationen

```
append([], L, L).
append([J|K], L, [J|KL]) :- append(K, L, KL).
```

```
J|K = [1,2,3]                    K = [2,3]
L = [4,5]              <-        L = [4,5]
J|KL = [1,2,3,4,5]               KL = [2,3,4,5]
```

CARL VON OSSIETZKY universität OLDENBURG

# Relationen

```
append([], L, L).
append([J|K], L, [J|KL]) :- append(K, L, KL).
```

```
position(X, [X|_], 0).
position(X, [_|L], P) :- position(X, L, Q), P is 1 + Q.
```

**contains(element, list)**

CARL VON OSSIETZKY universität OLDENBURG

# Relationen

```
append([], L, L).
append([J|K], L, [J|KL]) :- append(K, L, KL).
```

```
position(X, [X|_], 0).
position(X, [_|L], P) :- position(X, L, Q), P is 1 + Q.
```

```
contains(X, [X|_]).
contains(X, [_|L]) :- contains(X, L).
```

CARL VON OSSIETZKY universität OLDENBURG

# Städte, Länder und Kontinente

```
hauptstadtVon(london, england).
hauptstadtVon(madrid, spanien).
hauptstadtVon(kairo, aegypten).
hauptstadtVon(ottawa, kanada).
hauptstadtVon(rom, italien).
hauptstadtVon(washington, usa).

landInKontinent(england, europa).
landInKontinent(italien, europa).
landInKontinent(kanada, amerika).
landInKontinent(spanien, europa).
landInKontinent(aegypten, afrika).
landInKontinent(usa, amerika).
```

CARL VON OSSIETZKY universität OLDENBURG

# Städte, Länder und Kontinente

```
stadtInEuropa(Stadt) :- hauptstadtVon(Stadt, Land),
                        landInKontinent(Land, europa).
```

**stadtInKontinent(Stadt, Kontinent)**

CARL VON OSSIETZKY universität OLDENBURG

# Städte, Länder und Kontinente

```
stadtInEuropa(Stadt) :- hauptstadtVon(Stadt, Land),
                        landInKontinent(Land, europa).

stadtInKontinent(Stadt, Kont) :- hauptstadtVon(Stadt, Land),
                                 landInKontinent(Land, Kont).
```

CARL VON OSSIETZKY universität OLDENBURG

# Verbindungen

```
direct_connection(oldenburg, bremen).
direct_connection(oldenburg, delmenhorst).
direct_connection(delmenhorst, bremen).
direct_connection(bremen, hannover).

symmetric_connection(X,Y) :- direct_connection(X,Y).
symmetric_connection(X,Y) :- direct_connection(Y,X).
```

```
transitive_connection(X,Y)
```

CARL VON OSSIETZKY universität OLDENBURG

# Verbindungen

```
direct_connection(oldenburg, bremen).
direct_connection(oldenburg, delmenhorst).
direct_connection(delmenhorst, bremen).
direct_connection(bremen, hannover).

symmetric_connection(X,Y) :- direct_connection(X,Y).
symmetric_connection(X,Y) :- direct_connection(Y,X).

transitive_connection(X,Y) :- symmetric_connection(X,Y).
transitive_connection(X,Y) :- symmetric_connection(X,Z),
                              transitive_connection(Z,Y).
```

CARL VON OSSIETZKY universität OLDENBURG

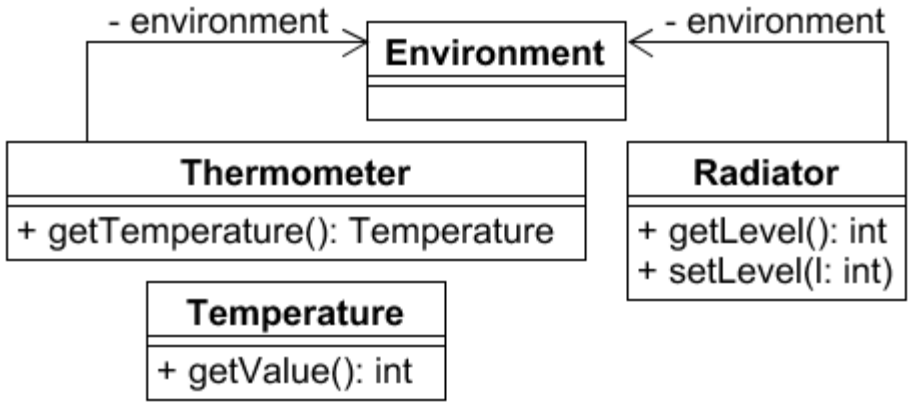
# Drools

CARL VON OSSIETZKY universität OLDENBURG

# Smart Home

- Internet of Things (IoT) Anwendung
- Sensoren
  - hier: Thermometer
- Aktoren
  - hier: Heizung
- Regeln
  - hier:
    - wenn es kälter als 18° ist, stelle die Heizung auf Stufe 5
    - wenn es wärmer als 22° ist, stelle die Heizung auf Stufe 0

# Smart Home: Architektur



CARL VON OSSIETZKY universität OLDENBURG

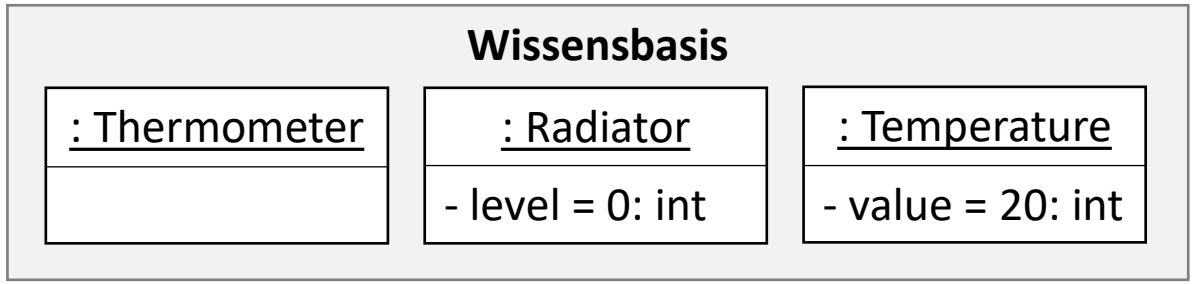
# Smart Home: Anwendung

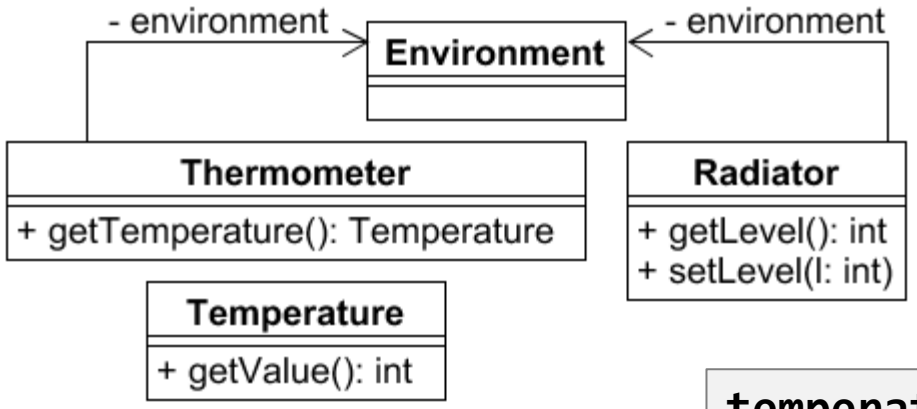
```
KieServices kieServices = KieServices.Factory.get();
KieContainer kContainer = kieServices.getKieClasspathContainer();
KieSession kSession = kContainer.newKieSession("ksession-rules");

Environment env = new Environment();
Thermometer thermometer = new Thermometer(env);
Radiator radiator = new Radiator(0, env);
kSession.insert(thermometer);
kSession.insert(radiator);
kSession.insert(thermometer.getTemperature());

kSession.fireAllRules();
kSession.dispose();
```

# Smart Home: Aufgabe




```
temperature < 18 ➔ radiator.setLevel(5)

temperature > 22 ➔ radiator.setLevel(0)
```

**Hinweis**: Die vom Thermometer gemessene Temperatur verändert sich mit der Zeit (d.h. es werden **`Temperature`**-Objekte mit anderen Werten zurückgegeben). Diese Veränderung muss entsprechend abgefragt werden.

CARL VON OSSIETZKY universität OLDENBURG

# Smart Home: Regeln

```
rule "New Temperature"
when
    $temp : Temperature()
    $thermometer : Thermometer()
then
    delete($temp);
    insert($thermometer.getTemperature());
end
```

CARL VON OSSIETZKY universität OLDENBURG

# Smart Home: Regeln

```
rule "Too cold"
salience 10
when
    Temperature(value < 18)
    $radiator : Radiator(level < 5)
then
    modify($radiator) {
        setLevel(5);
    }
    System.out.println("Set HeatingLevel to 5");
end
```

CARL VON OSSIETZKY universität OLDENBURG

# Smart Home: Regeln

```
rule "Too hot"
salience 10
when
    Temperature(value > 22)
    $radiator : Radiator(level > 0)
then
    modify($radiator) {
        setLevel(0);
    }
    System.out.println("Set HeatingLevel to 0");
end
```

# Smart Home: Erweiterung

- Wenn es kälter als 15° wird, soll die Heizung auf Stufe 10 gestellt werden
- Wenn es wärmer als 30° wird, soll eine Klimaanlage eingeschaltet werden