Listen und Arithmetik

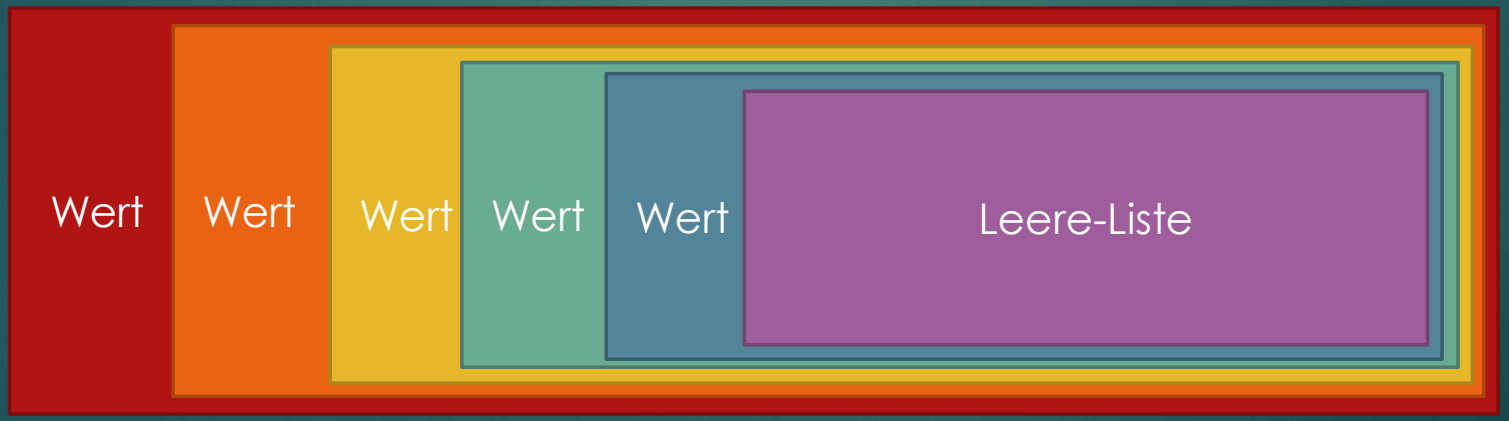
EIN VORTRAG VON FERDINAND TRENDELENBURG

Listen

LISTEN IN PROLOG UND JAVA

Listen

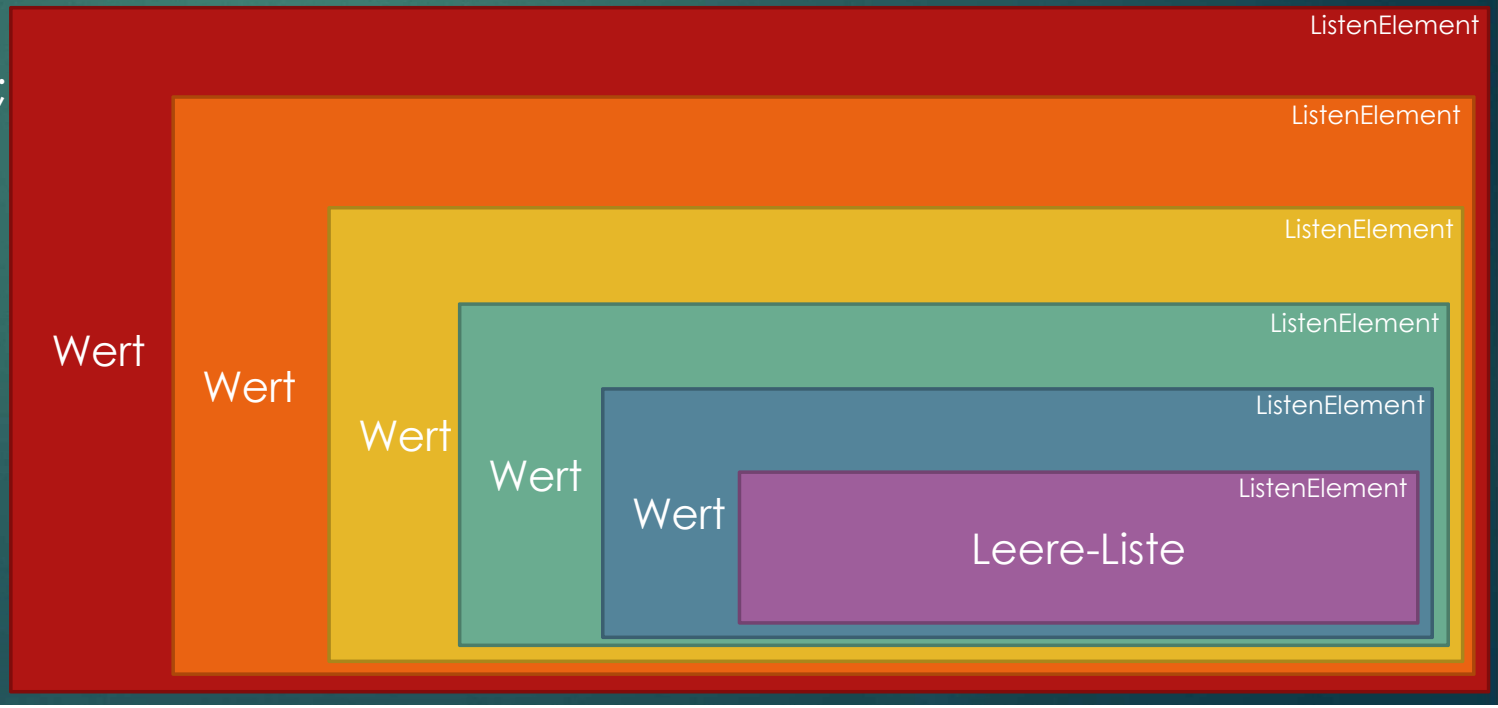
- Liste = Referenz auf das erste Listen-Element
- Listen-Element = [Wert | Rest]



[5, 4, 3, 2, 1, 0] == [5, [4, [3, [2, [1, [0, [ ]]]]]]]

Listen in Java

- Liste = Referenz auf das erste Listen-Element
- Objekt mit den globalen Variable:
  - Objekt Wert;
  - Listen-Element Rest;



Leere-Liste in Java

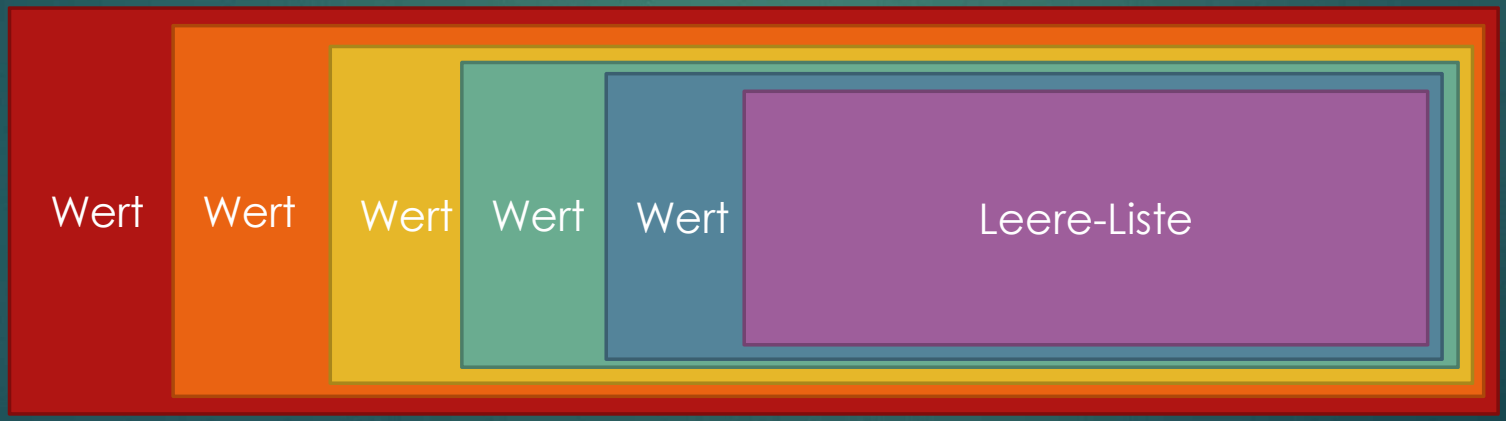
- Leere Liste / Leeres Listen-Element
- Konstruktor():
  - 1. nichts (leeres Listen-Element)
  - 2. Objekt und Listen-Element

```
public ListenElement() {
}


public ListenElement(Object data, ListenElement next) {
    this.data = data;
    this.next = next;
}
```

Listen in Prolog

- Listen-Element = Tupel(Wert, Listen-Element)



Leere-Liste in Prolog

- Leere Liste / Leeres Listen-Element
- Leeres Tupel = []

Arithmetik mit S-Zahlen

LISTEN IN PROLOG UND JAVA

Arithmetik mit S-Zahlen

- 0 = 0
- 1 = S(0)
- 2 = S(S(0))

S-Zahlen in Java

1
0

- globale Variablen:
  - S: next
- Konstruktor():
  - 1. nichts (0)
- setNext() ->
  - Nächstes Element erstellen

```
public void setNext() {
    this.next = new S();
}
```

0 in Java

- 0 != Integer sondern
- 0 == S() mit Konstruktor ohne Parameter
  - next = null
- isNull():

```
public boolean isNull() {
    return next==null;
}
```

- Optisch:

```
@Override
public String toString() {
    if(isNull()){
        return "0";
    }else {
        return "S(" + next.toString() + ")";
    }
}
```

S-Zahlen in Prolog

- 0 == 0 (Nummer)

```
nat(0).
nat(s(X)):-
  nat(X).
```

S(S(S(0)))

S(S(0))

S(0)

0