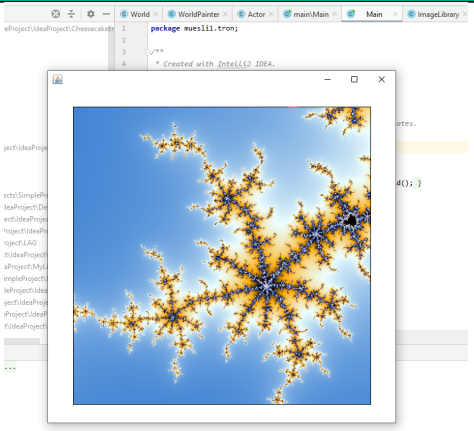
FOP Recap #7

Rekursion und Racket

TECHNISCHE UNIVERSITÄT DARMSTADT



Themen für heute

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweise

Objektorientiert oder Funktional?

Funktionen in Racket

Funktionen in Racket

Boolsche Logik

Verzweigungen

Rekursiv vs Iterativ

Listen

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hinweise
  - Integer-Division
  - Klammern
- Objektorientiert oder Funktional?
- Funktionen in Racket
- Funktionen in Racket
- Boolsche Logik
- Verzweigungen

Hinweise

Integer-Division

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
int a = 5 / 10;
int b = 21 / 20;
System.out.println(a);
System.out.println(b);
```

Hinweise
Integer-Division

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
int a = 5 / 10;
int b = 21 / 20;
System.out.println(a);
System.out.println(b);
```

```
$ 0
$ 1
```

- Achtung! Java Integer-Division!

Hinweise
Klammern

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  int a = (5 + 10) * 5;
2  int b = 5 + 10 * 5;
```

Hinweise
Klammern

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  int a = (5 + 10) * 5;
2  int b = 5 + 10 * 5;
```

```
$ 75
                $ 55
```

Hinweise
Klammern

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  int a = 5;
2  int b = (a == 5)? (a + 5) : -2;
```

Hinweise
Klammern

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  int a = 5;
2  int b = (a == 5)? (a + 5) : -2;
```

```
$ 5
                    $ 10
```

Objektorientiertes Design in Java

TECHNISCHE UNIVERSITÄT DARMSTADT

- Klassen stehen im Vordergrund
  - Und hierraus abgeleitete Objekte
- Objekte
  - Haben einen momentanen Zustand
- Methoden
  - Gehören immer zu einer Klasse

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweise

Objektorientiert oder Funktional?

Funktionen in Racket

Funktionen in Racket

Boolsche Logik

Verzweigungen

Rekursiv vs. Iterativ

Funktionales Design in Racket

TECHNISCHE UNIVERSITÄT DARMSTADT

- Funktionen stehen im Vordergrund
  - Können als Daten weitergegeben werden
  - Haben immer einen Rückgabewert
- Eine Funktion liefert mit denselben Parametern immer diesselbe Rückgabe
- Keine Variablen
- Keine statischen Typen

Achtung: Racket

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweise

Objektorientiert oder Funktional?

Funktionen in Racket

Funktionen in Racket

Boolsche Logik

Verzweigungen

Rekursiv vs. Iterativ

Funktionen in Racket

Deklaration und Aufruf

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  (define (my-function-name parameter second-parameter)
2      (+ parameter 1)
3  )
```

```
1  (my-function-name 25 #true)
```

```
1  (+ 25 23)
```

Funktionen in Racket

Vertrag

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  ;; Type: number ANY -> number
2  ;; Returns: The given number plus one
3  (define (my-function-name parameter second-parameter)
4      (+ parameter 1)
5  )
```

Konstanten

TECHNISCHE UNIVERSITÄT DARMSTADT

```
(define my-constant-name 28.25)
```

```
;; Type: number ANY -> number
;; Returns: The given number plus 28.25
(define (my-function-name parameter second-parameter)
    (+ parameter my-constant-name)
)
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweise

Objektorientiert oder Funktional?

Funktionen in Racket

Funktionen in Racket

Boolsche Logik

Verzweigungen

Rekursiv vs. Iterativ

Arithmetik

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  (+ 1 2)
2  (- 5 3)
3  (* 2 15)
4  (/ 18 5)
5  (modulo 9 4)
```

Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Präfixnotation
  - Operator vor Operand:
  - **Operator** Operand1 Operand2 .... OperandN
  - `+ 1 2 3 4 5`
- Infixnotation
  - Operator zwischen Operanden:
  - Operand1 **Operator** Operand2
  - `1 + 2 + 3 + 4 + 5`
  - `(((1 + 2) + 3) + 4) + 5`

Zahlen in Racket

TECHNISCHE UNIVERSITÄT DARMSTADT

- Können alles sein
  - Ganze Zahlen
  - Rationale Zahlen
  - Komplexe Zahlen
  - Nichtexakte Zahlen

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweise

Objektorientiert oder Funktional?

Funktionen in Racket

Funktionen in Racket

Boolsche Logik

Verzweigungen

Rekursiv vs. Iterativ

Boolsche Logik

TECHNISCHE UNIVERSITÄT DARMSTADT

- `true`
  - Auch `#true` oder `#t`
- `false`
  - Auch `#false` oder `#f`
- Und:
  - `and`
- Oder:
  - `or`
- Arithmetische Vergleichsoperatoren:
  - >, >=, <, <=, =

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweise

Objektorientiert oder Funktional?

Funktionen in Racket

Funktionen in Racket

Boolsche Logik

Verzweigungen

Rekursiv vs. Iterativ

Verzweigungen
mit `if`

TECHNISCHE UNIVERSITÄT DARMSTADT

```
(if condition expression-true expression-false)
```

```
(if (= number 5) -2 8)
```

```
(if (= number 5) (function-one #true) (function-two #false 5))
```

```
(if (= number 5)
    (function-one #true)
    (function-two #false 5)
 )
```

Verzweigungen
mit cond

TECHNISCHE UNIVERSITÄT DARMSTADT

```
(cond
    [condition-one expression-one]
    [condition-two expression-two]
    [....]
    [else expression-else]
)
```

```
(cond
    [(= number 5) #true]
    [(= number 2) 9]
    [....]
    [else #false]
)
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweise

Objektorientiert oder Funktional?

Funktionen in Racket

Funktionen in Racket

Boolsche Logik

Verzweigungen

Rekursiv vs Iterativ

Iterativer Ansatz

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
int[] result = new int[5];

for(int i = 0; i < 5; i++) {
    result[i] = i;
}
```

Rekursiver Ansatz

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void recursiveStep(int[] result, int currentIndex) {
    // Rekursionsanker
    if(currentIndex == result.length) {
        return;
    }
    result[currentIndex] = currentIndex;
    // Rekursiver Aufruf
    recursiveStep(array, currentIndex + 1);
}
```

```java
// Rekursions Start
int[] result = new int[5];
recursiveStep(result, 0);
```

Typische Rekursion in Racket

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  (define (recursive-function lst)
2      (cond
3          ; Rekursionsanker
4          [(empty? lst) empty]
5          [else (cons
6                  (+ 1 (first lst))
7                  ; Rekursiver Aufruf
8                  (recursive-function (rest lst))
9                )
10         ]
11     )
12 )
```

Typische Rekursion in Racket

TECHNISCHE UNIVERSITÄT DARMSTADT

```
(define (recursive-function lst)
    (cond
        [(empty? lst) empty]
        [else (cons
                (+ 1 (first lst))
                (recursive-function (rest lst))
              )
        ]
    )
)
```

```
; Rekursions Start
(recursive-function (list 1 2 3 4 5))
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweise

Objektorientiert oder Funktional?

Funktionen in Racket

Funktionen in Racket

Boolsche Logik

Verzweigungen

Rekursiv vs. Iterativ

Arrays in Java

TECHNISCHE UNIVERSITÄT DARMSTADT

- Arrays haben eine feste Länge
- Man kann auf einen beliebigen Index zugreifen

Listen in Racket

TECHNISCHE UNIVERSITÄT DARMSTADT

- Listen haben eine variable Länge
- Man kann nur auf
  - Das erste Element mit `(first ....)` zugreifen
  - Den Rest mit `(rest ....)` zugreifen
- Man erstellt sie über `(list element1 element2 .... elementN)`
- Man kann über `(empty? ....)` prüfen, ob sie leer ist
- Eine leere Liste erstellt man über `empty`
- Mit `(append lst1 lst2)` konkateniert man zwei Listen
- Mit `(cons element1 lst)` fügt man vorne an eine Liste ein neues Element an

Live-Coding!