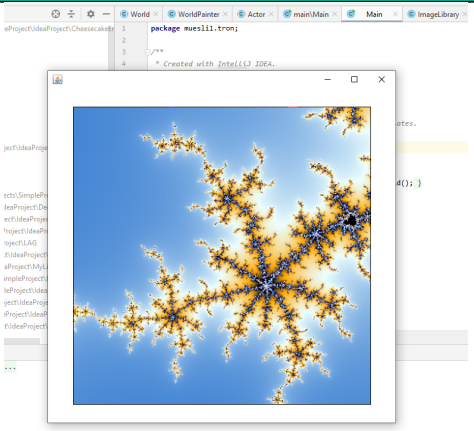
FOP Recap #6

Erste Schritte mit Racket



Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

- Organisatorisches
- Casting
- Integer-Division
- `String`
- Objektorientierung vs. Funktionales Design
- Racket

Themenwünsche

TECHNISCHE UNIVERSITÄT DARMSTADT

Abbildung: `https://pad.tudalgo.org/p/RX5J0xR5uv5vhP1clgHy`

- Schreibt hier eure Themenwünsche auf
- Auch Wiederholung von bereits besprochenen Themen möglich

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisatorisches

Casting
- Bei primitiven Datentypen
- Bei Objekttypen

Integer-Division

`String`

Objektorientierung vs. Funktionales Design

Racket

Casting

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bei primitiven Datentypen
  - um einen Zahlenwert in einem anderen Typen zu speichern
  - passiert implizit oder explizit
- Bei Objekttypen
  - um den statischen Typen zu ändern
  - kann jedoch je nach dynamischen Typen fehlschlagen

Casting

Bei primitiven Datentypen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jeder Zahltyp hat einen Bereich an Zahlen, den er repräsentieren kann
- `long`, `int`, `char`, `short`, `byte` für Ganzzahlen
- `double`, `float` für Komma-Zahlen
- Hierbei erkennt man folgende Rangordnung der Bereiche:
- `double` > `float` > `long` > `int` > `char` > `short` > `byte`

Casting

Bei primitiven Datentypen — Widening Casting

TECHNISCHE UNIVERSITÄT DARMSTADT

- Passiert implizit, automatisch

```
1  int a = 5;
2  int b = 27;
3  long c = a - b;
```

Casting

Bei primitiven Datentypen — Narrowing Casting

TECHNISCHE UNIVERSITÄT DARMSTADT

- Muss explizit angegeben werden, da Verlust von Präzision stattfinden kann!

```
1  int a = 288;
2  char c = a; // ERROR!
3  char c2 = (char) a; // OK!
4
5  char c3 = a + 5; // ERROR!
6  char c4 = ((char) a) + 5; // ERROR!
7  char c5 = (char) (a + 5); // OK!
```

Casting
Bei Objekttypen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Funktioniert nur ohne Fehler bei passenden dynamischen Typen
- Sonst wird eine Exception geworfen

```
1  class A { .... }
2  class B extends A { .... }
3  class C extends B { .... }
4
5  A a = ....;
6  C castedC = (C) a; // Nur möglich,wenn a dynamischen Typen von C
   ↪  oder Subtypen hat
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisatorisches

Casting

Integer-Division

`String`

Objektorientierung vs. Funktionales Design

Racket

Integer-Division

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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!

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

- Organisatorisches
- Casting
- Integer-Division
- `String`
  - Interner Aufbau
  - `char`
  - Beispiele
- Objektorientierung vs. Funktionales Design
- Racket

`String`
Interner Aufbau

TECHNISCHE UNIVERSITÄT DARMSTADT

- Im Prinzip nur ein `char[]`
- Jeder `String` ist unveränderbar

String
char

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ist wie `int` ein primitiver Datentyp
- Normale Verwendung:
  - Repräsentiert (im Normalfall) genau einen Buchstaben/Zeichen
  - Jedes Zeichen hat nach Unicode einen festen Zahlenwert zugeordnet

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  String s = "ABC";
2  char c0 = s.charAt(0); // == 'A'
3  char c1 = s.charAt(1); // == 'B'
4  char c2 = s.charAt(2); // == 'C'
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String s = "ABC";
char[] arr = s.toCharArray();
char c0 = arr[0]; // == 'A'
char c1 = arr[1]; // == 'B'
char c2 = arr[2]; // == 'C'
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  String s = "ABC";
2  String result = "";
3  result += s.charAt(2);
4  result += s.charAt(1);
5  result += s.charAt(0);
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String s = "ABC";
String result = "";
for(int i = s.length() - 1; i >= 0; i--) {
    result += s.charAt(i);
}
System.out.println(result);
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String s = "ABC";
String result = "";
for(int i = s.length() - 1; i >= 0; i--) {
    result += s.charAt(i);
}
System.out.println(result);
```

```
$ CBA
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- Für Groß und Kleinschreibung!

```java
char c = 'a';
char c2 = Character.toUpperCase(c); // == 'A'
boolean check = Character.isUpperCase(c2); // == true
char c3 = Character.toLowerCase(c2);
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  String message = "hallo";
2  String upperCaseMessage = message.toUpperCase();
3  System.out.println(upperCaseMessage);
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String message = "hallo";
String upperCaseMessage = message.toUpperCase();
System.out.println(upperCaseMessage);
```

```
$ HALLO
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisatorisches
Casting
Integer-Division
`String`
Objektorientierung vs. Funktionales Design
Racket

Bisher: 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

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
- Es gibt keinen Zustand
- Keine Variablen
- Fast alles ist ein Ausdruck
- Keine statischen Typen

Racket

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisatorisches

Casting

Integer-Division

`String`

Objektorientierung vs. Funktionales Design

Racket

- Funktionen in Racket
- Operatoren und Zahlen
- Verzweigungen

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)
)
```

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`

Arithmetik

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Zahlen in Racket

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Boolsche Logik

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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]
)
```

Live-Coding!