

# Informatik I - Zusammenfassung

1. Juni 2021

## Inhaltsverzeichnis

<b>1 Funktionen und Module</b>	<b>1</b>
1.1 Methoden . . . . .	1
1.1.1 Felder als Formalparameter . . . . .	1
1.1.2 Der Laufzeitstapel . . . . .	1

## 1 Funktionen und Module

### 1.1 Methoden

#### 1.1.1 Felder als Formalparameter

##### **Merke 1.1.1: Call-by-reference (Strukturierte Datentypen)**

Operationen mit Variablen des methodenaufrufenden Codes selbst.

**Strukturierte Daten** werden per **Referenz** übergeben! Änderungen der per Referenz übergebenen Parameter sind im **aufrufenden Code** sichtbar!

Beispiel: Einträge in übergebenen Feldern werden durch Methoden verändert! Das passiert bei reinen Integern (unstrukturierten Daten) nicht!

#### 1.1.2 Der Laufzeitstapel

##### **Merke 1.1.2: Definition**

Datenstruktur zur Verwaltung des Speichers der von den Methoden verarbeitet wird.  
**Speicherbereich, in dem lokale Variablen abgespeichert werden.**

##### **Merke 1.1.3: Stack Frame - Aktivierungsrahmen**

Abschnitt im Laufzeitstapel mit entsprechenden Speicherplätzen wird bei Aufruf **neu angelegt** und mit **[V,R]** initialisiert. Wird nach Wertübergabe (z.B. return) an Aufrufer freigegeben.

##### **Merke 1.1.4: Relevante Daten: V,R**

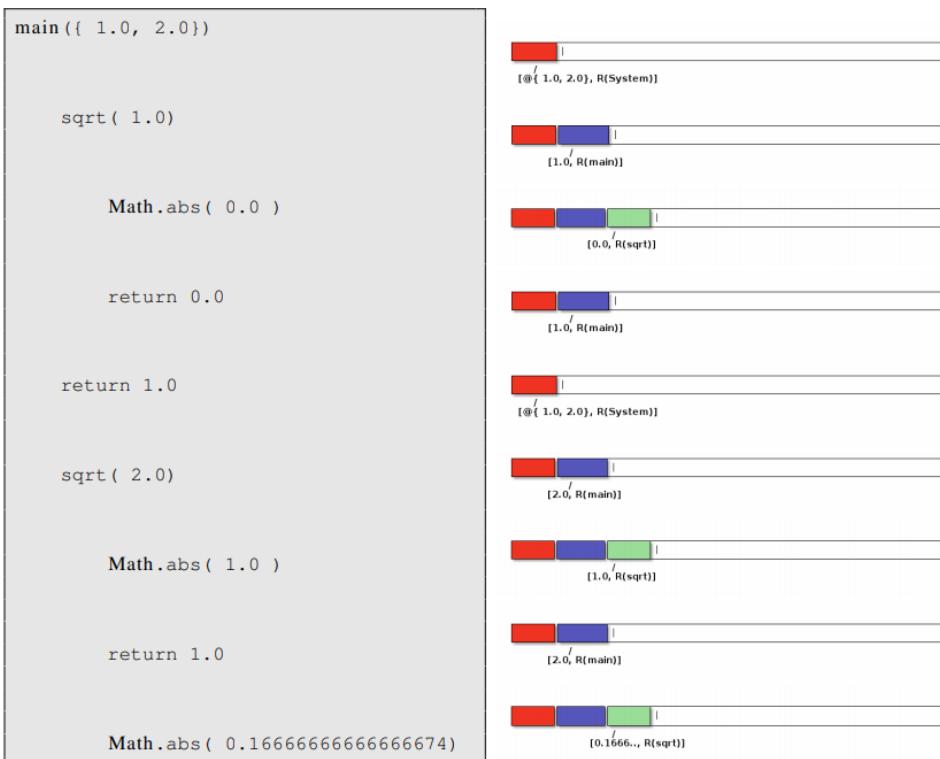
- **V:** Werte aller **lokalen Variablen** (Formalparameter + im Rumpf deklariert)

- R: **Rücksprungadresse** = Adresse des Befehls der dem Aufruf folgt (z.B. Vorherige Methode)

### Der Laufzeitstapel

= Speicherbereich, in dem lokale Variablen abgelegt werden

#### Methodentracing aus der Vogelperspektive



Analogie: Aktenstapel

### Merke 1.1.5: Rekursion

Wird durch den Laufzeitstapel ermöglicht.

**Rekursion** = sich selbst aufrufende Funktionen.

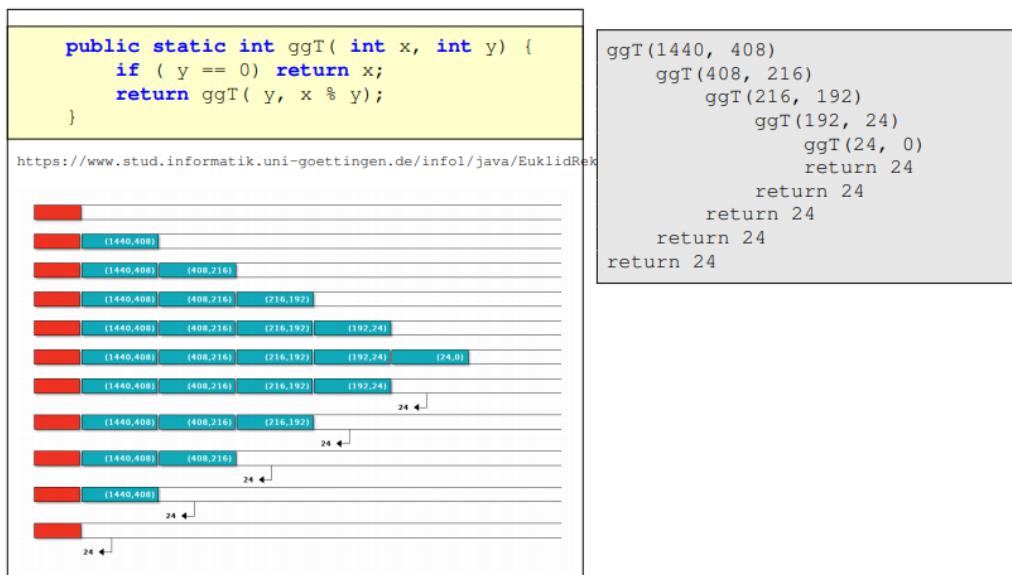
Beispiel: Fakultätsfunktion.

Grundmuster: **Initialisiere - Terminiere (falls trivial) - Rekurriere**

Jede Rekursion muss zu einem **Basisfall** führen. Sonst: STACKOVERFLOWERROR

- **Einfache Rekursion:** Lösung wird aus einem kleinen Vorgängerproblem gelöst. Geringe Aufwandreduktion pro Rekursionsschritt.
- **Mehrfahe Rekursion:** Gesamtaufwand gleichmäßig aufteilen. Verwende **Teile- und Herrsche- Algorithmus:**
  - Teile Problem in mehrere Unterprobleme ähnlicher Größe
  - Löse die Unterprobleme durch Teillösungen
  - Kombiniere die Teillösungen zur Problemlösung

## Beispiel: Berechnung des größten gemeinsamen Teilers



## Korrektheitsbeweis

- Für beliebige  $x \in \mathbb{N}$  gilt  $\text{ggT}(x, 0) = x$ .
- Ist  $y \neq 0$ , so gilt für jeden Teiler  $t$  von  $y$ :

$$t \text{ teilt } x \iff t \text{ teilt den Rest } r = x \bmod y,$$

- **Folglich** Ist  $y \neq 0$ , so  $\text{ggT}(x, y) = \text{ggT}(y, x \bmod y)$ .

### Merke 1.1.6: Rekursionstiefe

ist die Tiefe seines Rekursionsbaums (maximale Anzahl von Kanten auf einem Wurzel -> Blatt-Pfad)

### Merke 1.1.7: Traversierung

- **Inorder-Traversierung:** Links - Wurzel - Rechts (z.B. Scheiben)
- **Postorder-Traversierung:** Links - Rechts - Wurzel (z.B. Fibonacci)

### Merke 1.1.8: Memoisation

Wiederholungen vermeiden durch **zwischenspeichern** in einem Feld.

Beispiel Fibonacci:

```
static int[] f = new int[N];
public static void mFib() {
    f[0] = 0; f[1] = 1;
    for (int i = 2; i < N; ++i)
        f[i] = f[i-1] + f[i-2];
}
```

### Wichtig 1.1.1: Rekursion

- Es wird ein Basisfall benötigt!
- Wir müssen die Problemgröße durch Rekursion reduzieren.
- Der Speicherbedarf darf nicht zu groß werden.
- Wiederholte Berechnungen vermeiden!