Funktionen & Rekursion

Inhaltsverzeichnis

TECHNISCHE UNIVERSITÄT DARMSTADT

Erste Funktionen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel1 {
        public static void main(String[] args) {
                //Die Schönheit von Redundanz?
                System.out.println("Ich␣bin␣redundant");
                System.out.println("Ich␣bin␣redundant");
                System.out.println("Ich␣bin␣redundant");
        }
}
```

Frage

Was macht das Programm?

Erste Funktionen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel1 {
        public static void main(String[] args) {
                //Die Schönheit von Redundanz?
                System.out.println("Ich␣bin␣redundant");
                System.out.println("Ich␣bin␣redundant");
                System.out.println("Ich␣bin␣redundant");
        }
}
```

Frage

Was macht das Programm?

Aber

...was wenn ein anderen String ausgeben werden soll?

Erste Funktionen
Die Idee hinter Funktionen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Funktionen können diverse Funktionalitäten kapseln, die häufiger benötigt werden
- Diese Funktionalität wird an einem Ort zentriert gespeichert
  - Keine **Redundanz** mehr
- Möchte man sie nutzen verweist man nur auf diese eine Stelle

Hinweis

Einige Funktionen haben wir schon kennen gelernt, beispielsweise System.out.println(`"Hello World"`) und public static void main(String[] args)

Erste Funktionen
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel2 {
        public static void main(String[] args) {
                sagHallo();
                sagHallo();
                sagHallo();
        }

        public static void sagHallo() {
                System.out.println("Hello World");
        }
}
```

Aus was besteht eine Funktion?

TECHNISCHE UNIVERSITÄT DARMSTADT

Eine **Funktion** besteht aus:

- Einem Funktionskopf, der sich aus (**ggf Modifikatoren**,) **Rückgabetyp**, **Funktionsname** und **Parametern** zusammensetzt

```
1 public static void sagHallo()
```

- Einem Funktionsrumpf, der beschreibt was die Funktion macht

```
1 {
2         System.out.println("Hello World");
3 }
```

Fallbeispiel: Währungsrechner

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konvertierung von **Dollar** in **Euro**

```
public class Beispiel3 {
        public static void main(String[] args) {

                double dollar = 20.5;
                System.out.println(dollar + "␣Dollar␣sind␣" +
                    dollar * 0.729394602 + "␣Euro");
        }
}
```

Problem

Was wenn sich der Kurs häufig ändert? $\Rightarrow$ Viele Änderungen

Fallbeispiel: Währungsrechner Verbesserung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Erste Verbesserung ist den Kurs in eine Variable auszulagern

```
public class Beispiel4 {
        public static void main(String[] args) {

                double dollar = 20.5;
                double exchangeRate = 0.729394602;
                System.out.println(dollar + "␣Dollar␣sind␣" +
                    dollar *  exchangeRate + "␣Euro");
        }
}
```

Problem

Sobald man mehr als eine Währungsumwandlung hat, wird es unübersichtlich!

Fallbeispiel: Währungsrechner
Parameter

TECHNISCHE UNIVERSITÄT DARMSTADT

Lösung

Funktion, welche den Betrag als **Parameter** erwartet und die Konvertierung ausgibt

- Analog zu Parametern einer mathematischen Funktion: $f(x) = 2x$
- **Abstrakt**

```
1 Rückgabetyp Funktionsname(type1 name1, type2 name2,....)
```

- **Konkret**

```
1 //Deklaration des Funktionkopfes
2 void funktion(double value1, int value2, String value3) {
3         //In der Funktion Parameter wie Variablen verweden
4         double erg = value1 + value2;
5 }
6 funktion(20.0,22,"The Word Alive - Entirety"); //Aufruf
```

Parameter
Parameter einer Funktion übergeben

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel5 {
        public static void main(String[] args) {
                double dollar = 20.5;
                convertDollarToEuro(dollar);
        }
        public static void convertDollarToEuro(double amount) {
                double exchangeRate = 0.729394602;
                System.out.println(amount + "␣Dollar␣sind␣" +
                    amount *  exchangeRate + "␣Euro");
        }
}
```

Problem

Aber was wenn man innerhalb der Main-Funktion nun mit der neuen Währung weiter rechnen möchte?

Die return Anweisung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Mit der **return** können Werte aus einer Funktion zurückgegeben werden
- Der Datentyp den man zurück gibt muss mit dem Rückgabewert übereinstimmen!

```
1  public class Primzahl {
2          public static void main(String[] args) {
3                  int number = 666;
4                  boolean isNumberPrim = isPrim(number);
5                  //...
6          }
7          public static boolean isPrim(int primNumber) {
8                  for(int i = 2; i < primNumber; i++)
9                          if(primNumber % i == 0) return false;
10                 return true;
11         }
12 }
```

Die return Anweisung
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel6 {
        public static void main(String[] args) {
                double dollar = 20.5;
                System.out.println(dollar + "␣Dollar␣sind␣" +
                    convertDollarToEuro(dollar) + "␣Euro");
        }
        public static double convertDollarToEuro(double amount) {
                double exchangeRate = 0.729394602;
                return amount * exchangeRate;
        }
}
```

Die return Anweisung
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
// ...
public static double convertDollarToEuro(double amount) {
        double exchangeRate = 0.729394602;
        return amount * exchangeRate;
        System.out.println("Ich␣werde␣nie␣ausgeführt");
}
// ...
```

Achtung

Anweisungen nach dem return Statement werden nicht mehr ausgeführt

Die return Anweisung
Veränderung nur an einer Stelle

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vorteil durch die Kapselung der Funktionalität ist das sich Änderungen nur an einer Stelle auswirken
- Die Bank behält 2 % des Betrages ein

```
1 public class Beispiel9 {
2         public static void main(String[] args) {
3                 convertDollarToEuro(20.5);
4         }
5 public static double convertDollarToEuro(double amount) {
6                 double exchangeRate = 0.729394602;
7                 return amount * exchangeRate * 0.98;
8         }
9 }
```

Hinweis:

Es muss nur die Funktion selbst angepasst werden, nicht alle Aufrufe der Funktion

Arrays an Funktionen übergeben

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class ArrayBeispiel {
        public static void main(String[] args) {

                int[] array = {1,2,3,4,5,6,7,8,9,10};
                printAll(array);
        }

        public static void printAll(int[] werte) {
                for(int wert : werte) {
                        System.out.print(wert + ",");
                }
        }
}
```

Beliebig viele Parameter übergeben

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class VieleParameter {
        public static void main(String[] args) {

                funktion(1,2,3,4,5);
                funktion(6,7,8,9);
        }

        public static void funktion(int ... viele) {
                System.out.print(viele[0] + ",␣");
                System.out.print(viele[1] + ",␣");
                System.out.print(viele[2] + ",␣");
                System.out.print(viele[3] + ",␣");
        }

}
```

Beliebig viele Parameter übergeben

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class VieleParameter2 {
        public static void main(String[] args) {

                funktion(1,2,3,4,5);
                funktion(6,7,8,9);
        }

        public static void funktion(int ... viele){
                for(int wert : viele) {
                        System.out.print(wert + ",␣");
                }
        }

}
```

Rekursion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Funktionen können sich selbst aufrufen
- Solange bis eine Funktion einen Rückgabewert liefert

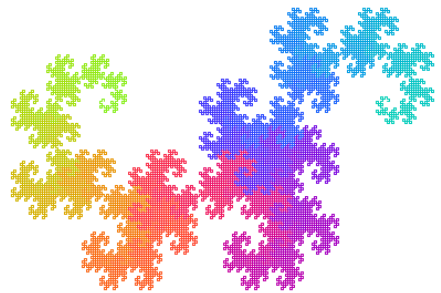

Abbildung: Drachenkurve

Achtung

Mit Bedacht einsetzen, viel Speicher kann verwendet werden

Rekursion
Erstes Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel10 {
        public static void main(String[] args) {
                count(0);
        }

        public static void count(int value) {
                System.out.println(value);
                count(value + 1);
        }
}
```

Frage

Was tut das Programm, das eine rekursive Funktion implementiert hat?

Rekursion
Erstes Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel10 {
        public static void main(String[] args) {
                count(0);
        }

        public static void count(int value) {
                System.out.println(value);
                count(value + 1);
        }
}
```

Frage

Was tut das Programm, das eine rekursive Funktion implementiert hat?

$\Rightarrow$ Es zählt bis ein paar tausend und stürzt schließlich ab!

Rekursion
Der Rekursionsanker

TECHNISCHE UNIVERSITÄT DARMSTADT

- Es wird ein Abbruchkriterium benötigt

```
1  public class Beispiel11 {
2          public static void main(String[] args) {
3                  count(0);
4          }
5
6          public static void count(int value) {
7                  System.out.println(value);
8                  if(value < 1000) { //Der Rekursionsanker
9                          count(value + 1);
10                 }
11         }
12 }
```

Rekursion
Der Rekursionsanker

TECHNISCHE UNIVERSITÄT DARMSTADT

- Es wird ein Abbruchkriterium benötigt

```
1  public class Beispiel11 {
2          public static void main(String[] args) {
3                  count(0);
4          }
5
6          public static void count(int value) {
7                  System.out.println(value);
8                  if(value < 1000) { //Der Rekursionsanker
9                          count(value + 1);
10                 }
11         }
12 }
```

- Jetzt werden die Zahlen 0 bis 1000 ausgegeben

Rekursion
Summe

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgabe: Die Summe der ersten **n** Zahlen bestimmen

- Die Funktion $sum : \mathbb{N}_0 \to \mathbb{N}_0$ ist dabei wie folgt definiert: $sum(n) = \sum_{i=0}^{n} i$
- Rekursive Definition extrahieren
  - Trivial Fall bzw. Rekursionsanker: sum(0) = 0
  - Rekursionsschritt ($n > 0$): Zurückführen auf einfacheren Fall (n-1)
- Die Summe der ersten n Zahlen lässt sich nun durch: $sum(n) = sum(n-1) + n$ berechnen
- Es folgt für die Rekursive Definition: $sum(n) = \begin{cases} 0 & \text{, falls } n = 0 \\ sum(n-1) + n & \text{, sonst} \end{cases}$

Rekursion
Rekursionsbaum

TECHNISCHE UNIVERSITÄT DARMSTADT

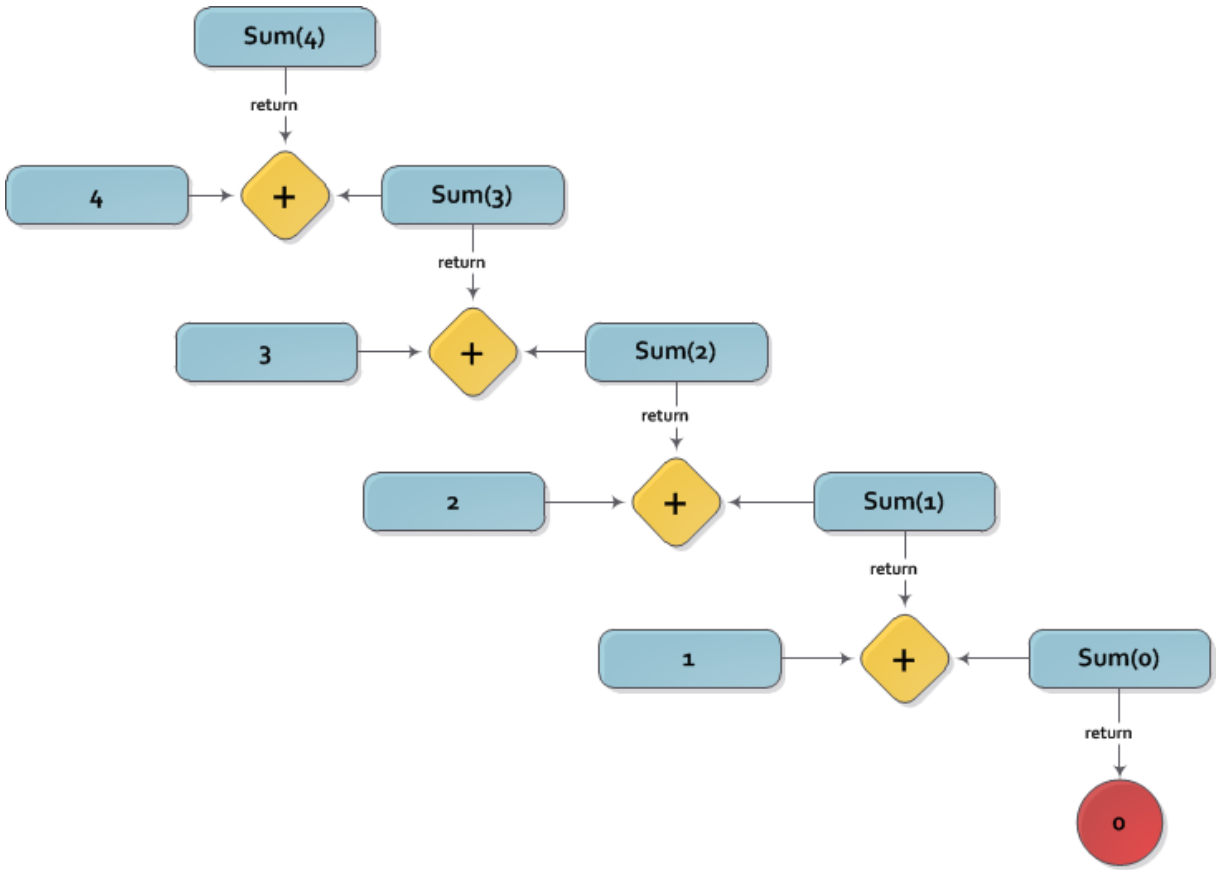

Abbildung: Rekursionsbaum

Rekursion
Rekursive Summe in Java

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel12 {
        public static void main(String[] args) {
                System.out.println(sum(20));
        }

        public static int sum(int value) {

                if(value == 0) {
                        return 0;
                }
                return value + sum(value -1);
        }
}
```

Hinweis

Jede Rekursive Funktion kann man als iterative Funktion implementieren

Rekursion
Iterative Summe in Java

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Beispiel13 {
        public static void main(String[] args) {
                System.out.println(sum(20));
        }

        public static int sum(int value) {
                int erg = 0;
                for(int i=0;i<=value;i++) {
                        erg += i;
                }
                return erg;
        }
}
```

Formales rekursives addieren in $\mathbb{N}$

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lasst euch hier überraschen ;)

Quellen

TECHNISCHE UNIVERSITÄT DARMSTADT

- **[1]** http://blog.rki-home.de/wp-content/uploads/2011/05/drachenkurve_14.png