Java Einführung
Methoden

Kapitel 6

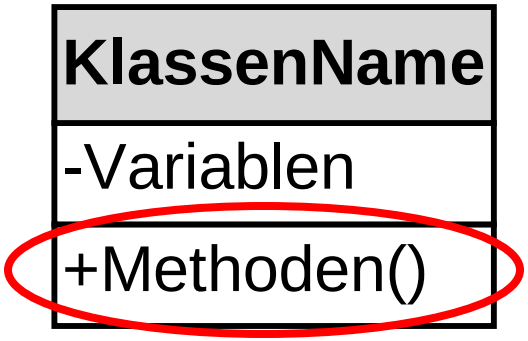
Inhalt

- Deklaration und Aufruf von Methoden
- Lokale und globale Namen (Bezeichner)
- Sichtbarkeit und Lebensdauer von Variablen in Methoden
- Überladen von Methoden

Methoden

- Methoden gehören logisch zusammen und lösen einen bestimmte Teilaufgabe.
- Methoden werden zur Lösung der Teilaufgabe aufgerufen.

- In Klassen werden Methoden eingesetzt um das Verhalten der Klasse abzubilden.



Deklaration von Methoden

- Methoden haben eine **Namen (Bezeichner)** wie Variablen.
- Methoden haben ***Parameter***. Das sind Werte, die beim Aufruf einer Methode übergeben werden.

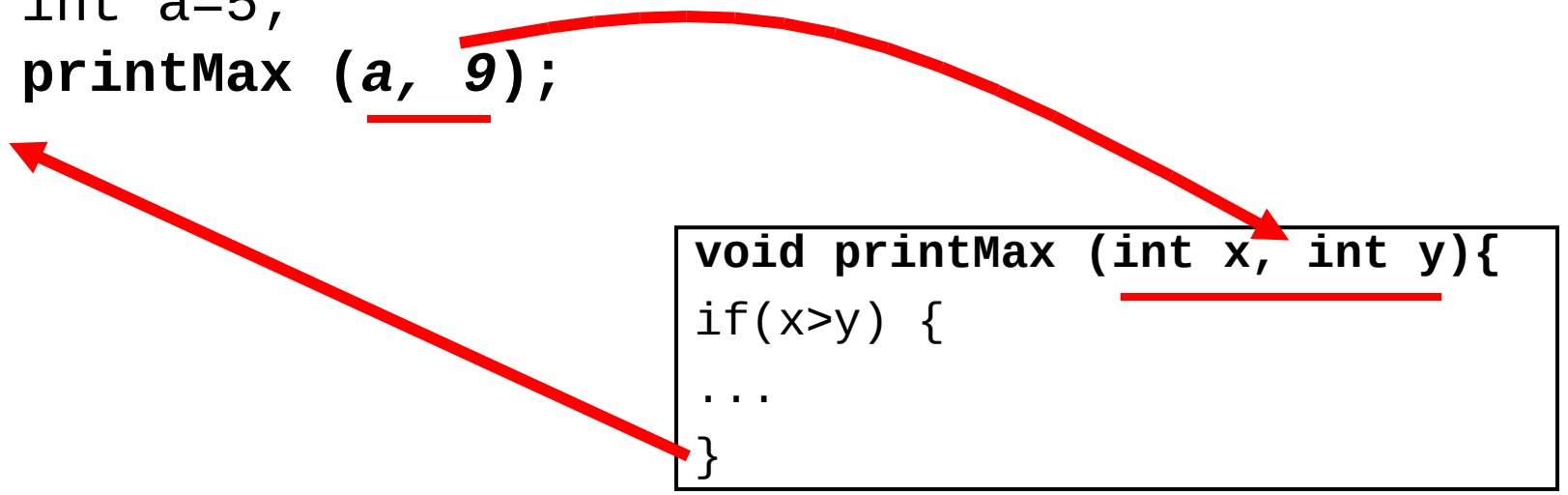
```
static void printMax (int x, int y) {
  if (x>y) {
    System.out.print(x);
  } else {
    System.out.print(y);
  }
}
```

Signatur

Aufruf von Methoden

Durch Angabe von Name und Parameter der Methode:

```
public static void main (String[] arg) {
  int a=5;
  printMax (a, 9);
}
```



Beim **Aufruf** wird den *formellen Parametern (x, y)* der Wert der *aktuellen Parametern (a, 9)* **zugewiesen**. x und y können nur in der Methode printMax verwendet werden.

Deklaration von Methoden mit Rückgabewert

```
static int max (int x, int y) {
  if (x>y) {
    return x;
  } else {
    return y;
  }
}
```

- Der Rückgabewert ist hier als `int` definiert.
- `return` legt den Rückgabewert fest und verlässt die Funktion.
- Es kommen alle Datentypen für den Rückgabewert in Frage. Das Schlüsselwort `void` bedeutet: kein Rückgabewert

Aufruf von Methoden mit Rückgabewert

Durch Angabe von Name und Parameter der Methode:

```
public static void main (String[] arg) {
   int a=15; int maximum;
   maximum = max(a, 9);
   System.out.println(maximum);
}
```

```
int max (int x, int y){
if(x>y) { return x; }
else { return y; }
}
```

Bei der Rückgabe wird der Rückgabewert (Wert von `x`, 15) als Ergebnis der Funktion `max(15,9)` der Variable `maximum` **zugewiesen**.

Beispiele: Aufrufen von Methoden

```
// Def. der Methode max von der vorherigen Folie

int a=15; int m;

m = max(a/5,9);

m = max(a,7) + 19

if (max(-a,23) == 35) { ... }
```

Rückgabewerte

- Als Rückgabewert ist ein **einzelnes Element** eines Datentyps (auch eine Instanz) möglich.
  z.B.:
  ```
  int addiere(int a, int b) {...}
  ```

- Hat die Methode keinen Rückgabewert, wird das Schlüsselwort **`void`** verwendet.
  z.B.:
  ```
  void ausgeben(String text) { ...}
  ```

Lokale Namen

- **Lokale Namen** (Methoden, Variablen) sind nur in der Methode (im Block) verfügbar, in der sie deklariert wurden

```
static void macheWas(int x) {
   int y;
       // x und y sind nur in dieser Methode verfügbar
   ...
}
```

Gültigkeitsbereich von Variablen

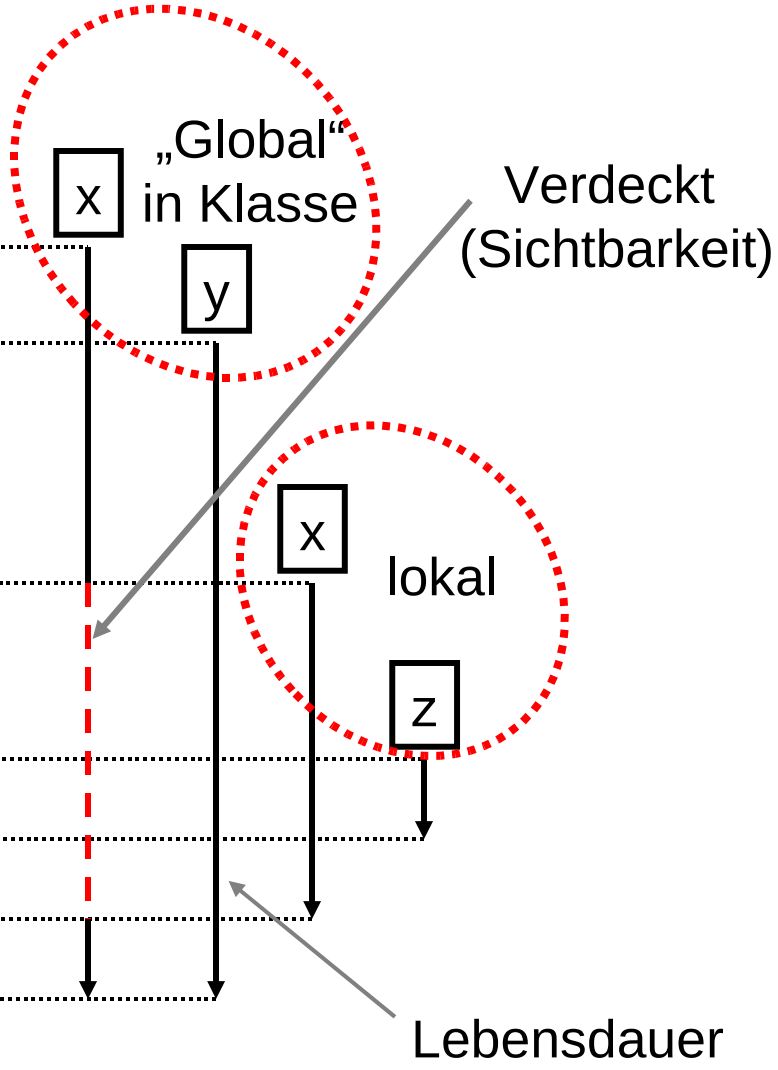
- **Gültigkeitsbereich:** ist jener Bereich im Programm, in dem auf eine Variable über den Namen zugegriffen werden kann

  - **Sichtbarkeit:** Block (+ Unterblöcke), in dem die Variable deklariert wurde.

  - **Lebensdauer:** werden bei betreten des Blocks angelegt und bei Verlassen wieder freigegeben.

Bsp: Sichtbarkeit und Lebensdauer

```
class Prog {
  static int x;
  static int y;
  ...
  static void machen(){
   int x;
   while(...) {
      int z;
   }
  }
}
```



Überladen von Methoden

- Allgemein gilt: In einem Block deklarierte Namen müssen unterschiedlich sein. Bei Methoden gibt es eine Ausnahme:

Zwei Methoden dürfen den gleichen Namen haben, wenn sich durch ihre Parameterliste (Art und Anzahl) unterscheidet. ***Rückgabewert und Name der Parametervariablen sind egal beim Überladen egal!***

```
static void print (int x) { ... }
static void print (double x) { ... }
static void print (int x, int y) { ... }
```

Bsp: Aufruf überladener Methoden

```
class MaxProgramm {
   static int max (int x, int y) {
     if (x>y) { return x; } else { return y; }
   }

   static int max (int x, int y, int z) {
     int max;
     max = max(z, max(x,y));
     return max;
   }

   public static void main (String[] arg) {
     System.out.println(max(45,13,98));
   }
}
```

Die richtige Methode wird beim Aufruf über die **Signatur** automatisch ausgewählt.

Fehler beim Überladen von Methoden

```
1. static int addiere (int a, int b) {...}
2. static int addiere (int a, int b, int c) {...}
3. static double addiere (double a, double b) {...}
4. static double addiere (int x, int y) { ... }
```

Die Methoden in Zeile 1 und 4 können nicht gemeinsam deklariert werden, da beide die gleiche **Signatur** **`addiere(int,int)`** haben. Der Compiler bricht die Übersetzung mit einer Fehlermeldung ab.

Nach dieser Einheit sollten Sie ...

- Methoden deklarieren und aufrufen können
- Über die Sichtbarkeit und Lebensdauer von Variablen bescheid wissen
- Das überladene von Methoden kennen