Variablen

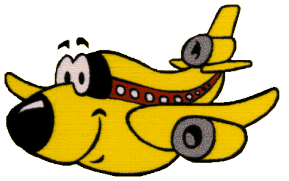
4213

int

17.834

float

Flugzeug



36812736294

long

"I write code"

String

Hund

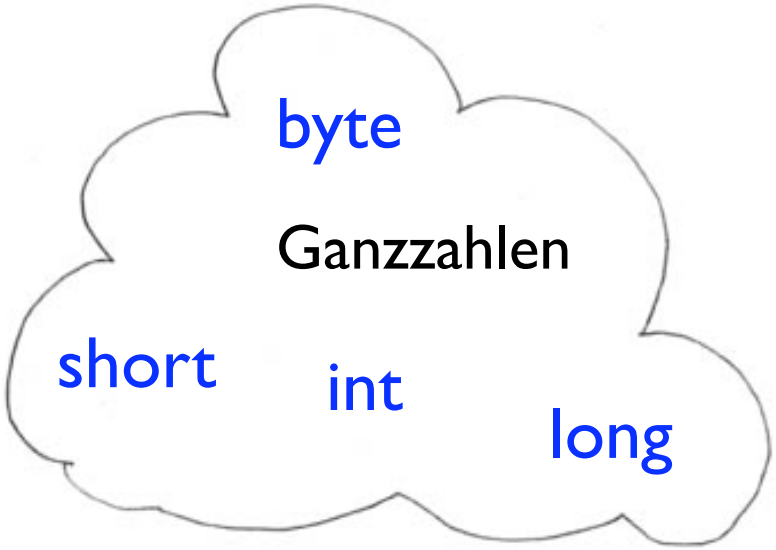
Variablen sind wie Becher.

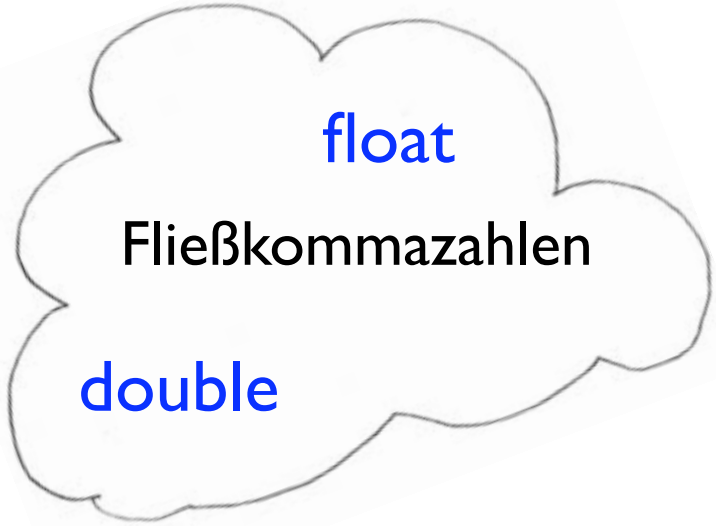
Sie speichern etwas.

// Variablen brauchen
// einen **Typ** und
// einen **Namen**

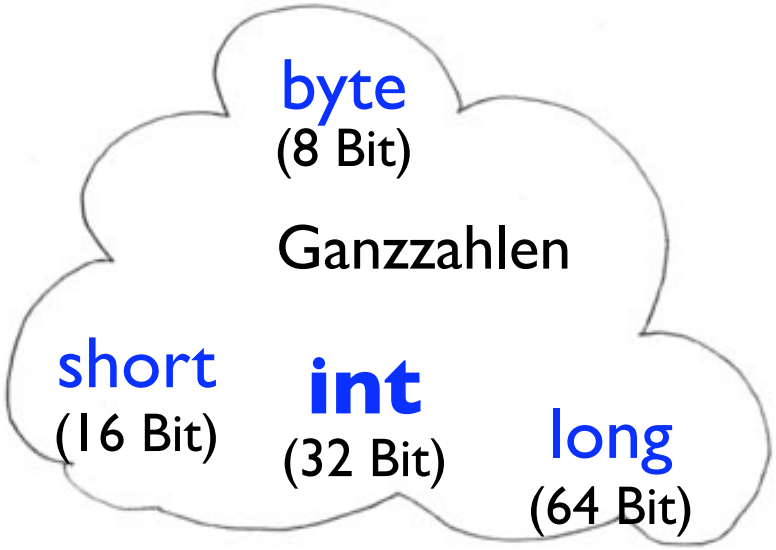
```
int gewicht;
String vorname;
```

Elementare Datentypen



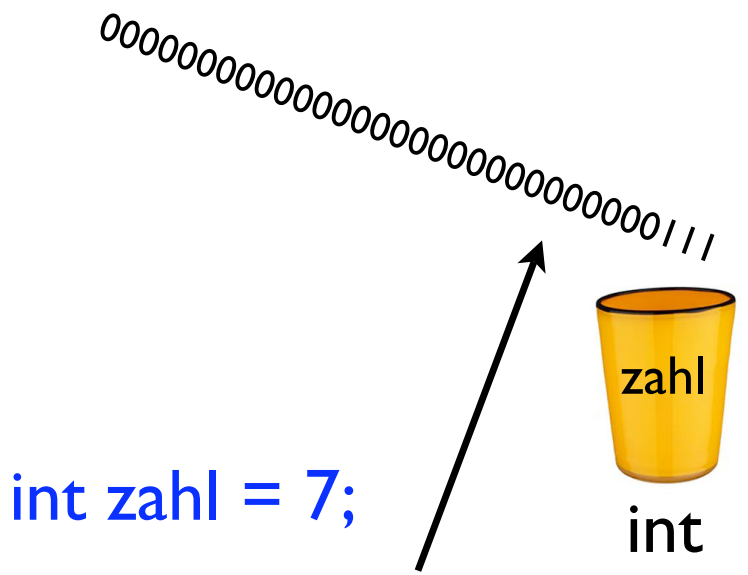


32 Bit speichern Zahlen von -2147483648 bis 2147483647

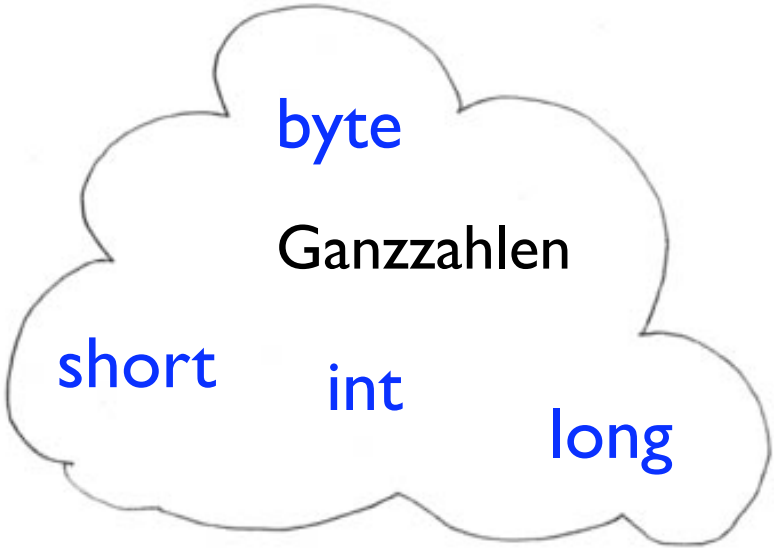


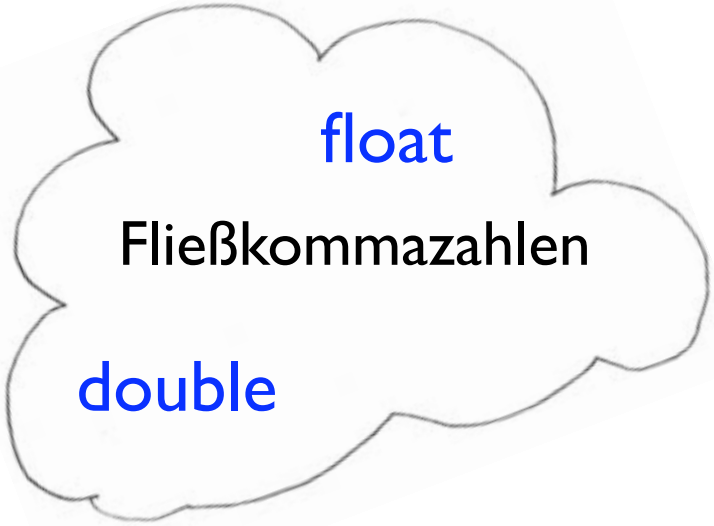
int temperatur = -20;

int distanz = 93743286;



Elementare Datentypen



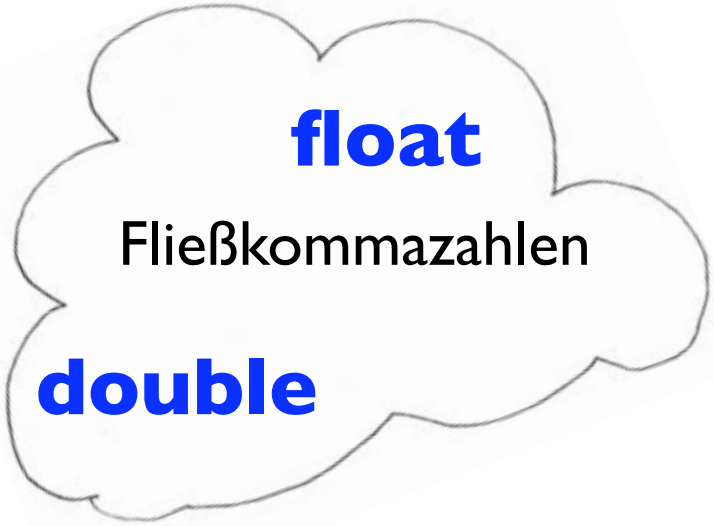


float preis = 137.99f;

32 Bit

Dezimalpunkt

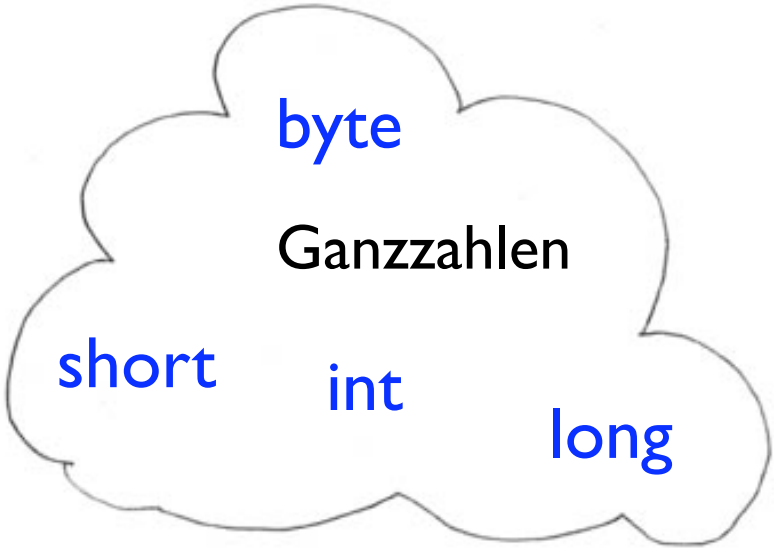
"f" identifiziert float

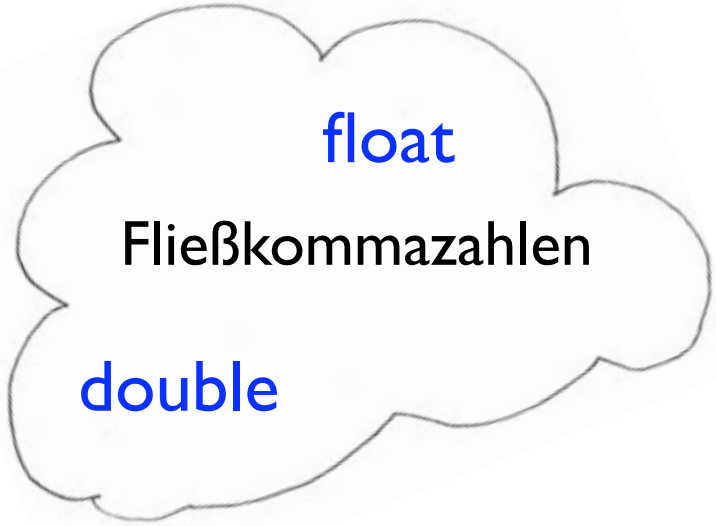


double gewicht = 14091.8;

64 Bit

Elementare Datentypen





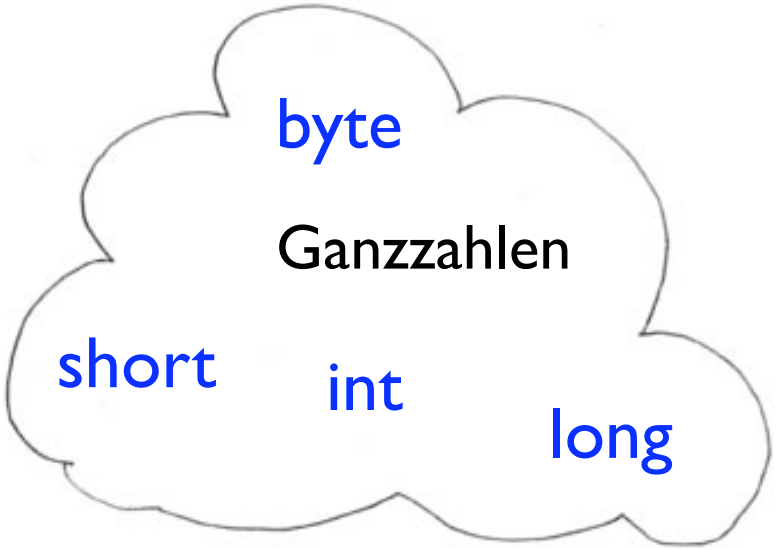
```
boolean istPause      = false;
boolean istVorlesung = true;
```

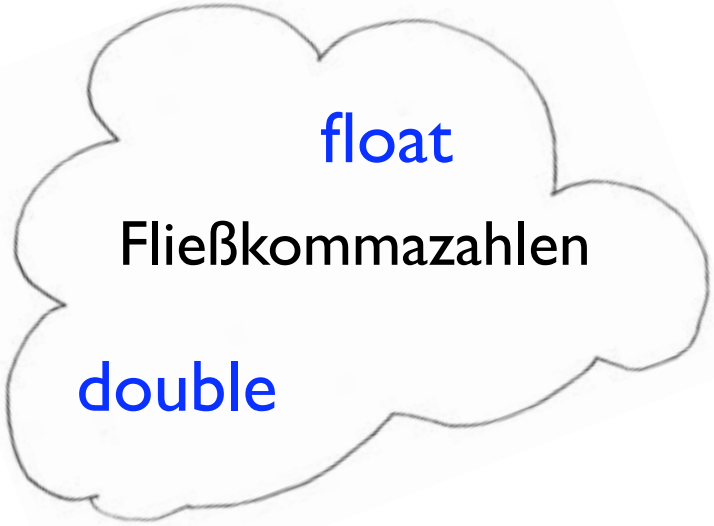
einfache Anführungszeichen

```
char zahl = 65;
char zeichen = 'A';
```

16 Bit: 0 bis 65535

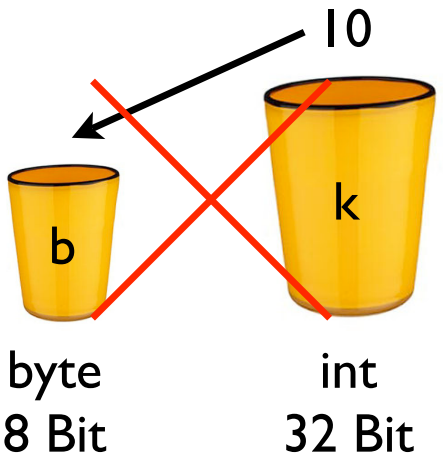
Elementare Datentypen





Spielen Sie Compiler! Finden Sie die Fehler?


```
int pi = 3.14159265;
float pi = 3.14159265;
long sehrWeit = 314159265
boolean istWahr = “true”;
```




```
int k = 10;  byte b = k;    // Geht nicht!
byte b = 10;  int k = b;    // Geht; implizite Typum-
                            // wandlung durch Compiler
```

Variablennamen

String artikel;

String _zutat1;

float $preis;

int anzahl_Gummibaerchen_in_Tuete;

void public short String main int break while class static if ...

Geht das?

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

    String artikel    = "Haribo Goldbaeren";
    String 2artikel   = "Haribo Colorado";
    String super      = "Die Echten ohne Fett";

    // 20000 Stück importieren
    short import      = 20000;
}
```

PS: **"literale Werte"** werden beim Compilieren definiert.

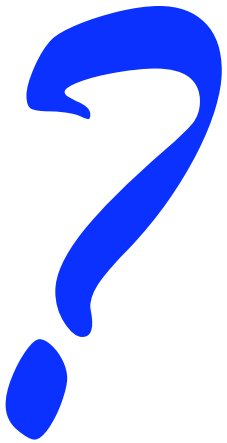
Geht das?

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

    String artikel    = "Haribo Goldbaeren";
    String 2artikel   = "Haribo Colorado";
    String super      = "Die Echten ohne Fett";

    // 20000 Stück importieren
    short import      = 20000;
}
```

Das wissen sie schon:

```
int anzahlStudentenInVorlesung = 696;
String mein_lieblings_essen = "Studentenpizza";
float preis = 3.10f;
```

Was läuft eigentlich hier ab?



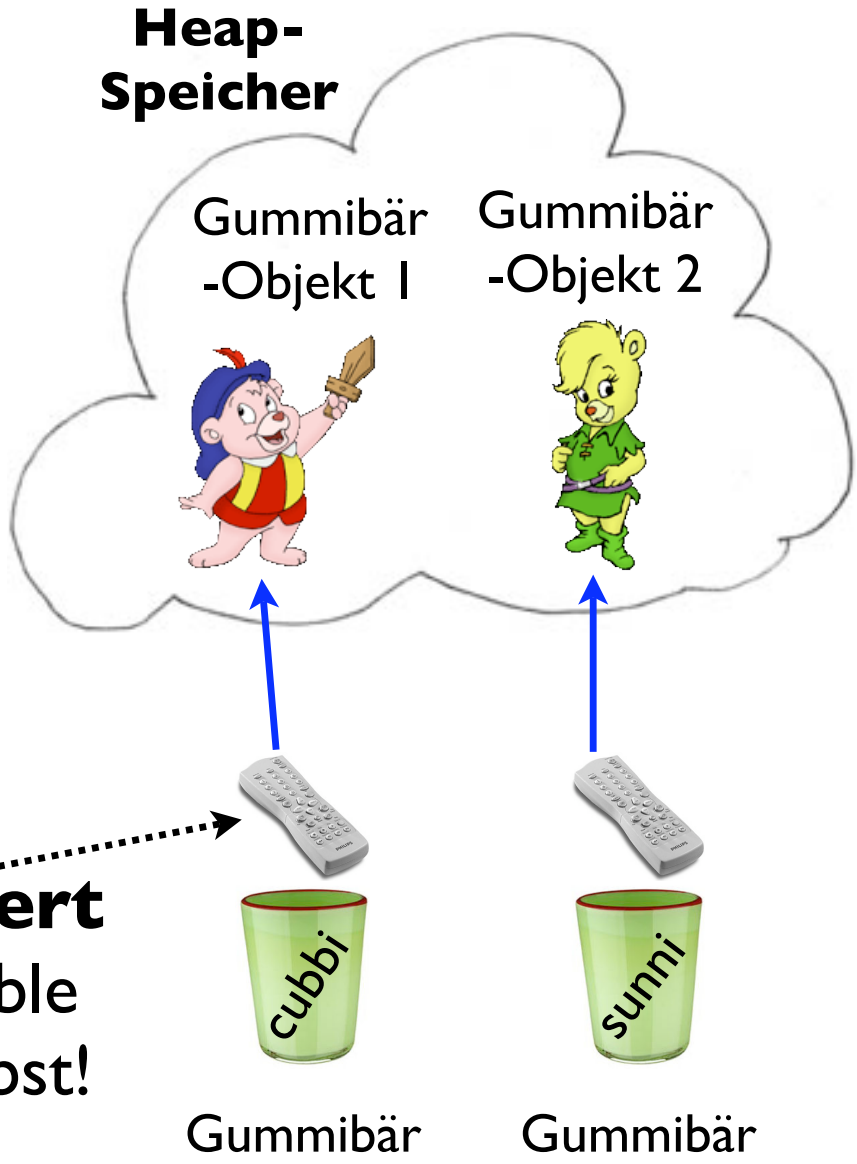
```
class Gummibär { // Gummibärcode... }
Gummibär cubbi;
cubbi = new Gummibär ();
```

Referenztypen

Gummibär cubbi;

cubbi = new Gummibär ();

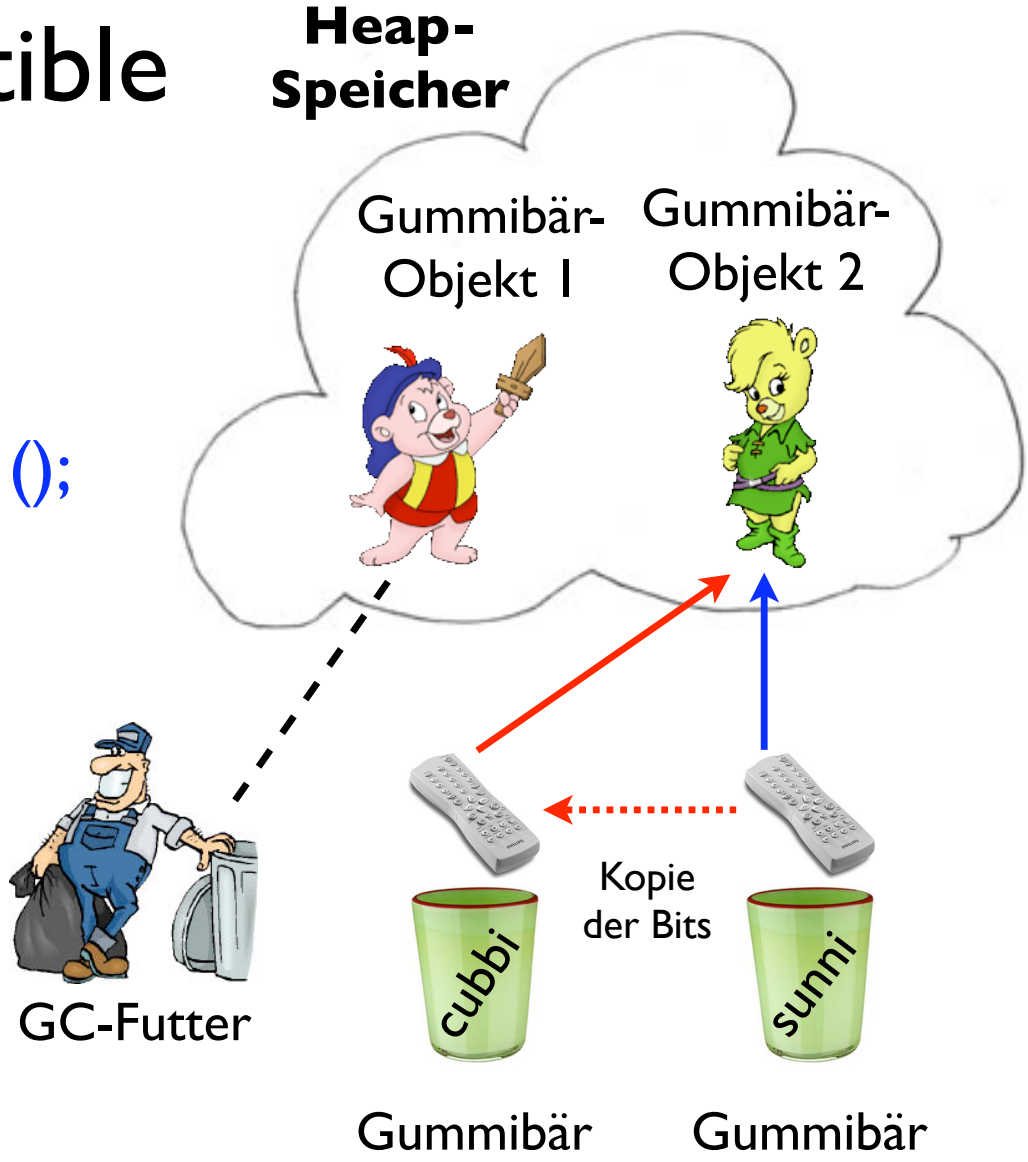
Gummibär sunni =
new Gummibär ();



Die Bits für den **Referenzwert** werden in die Referenz-Variable gesteckt, nicht das Objekt selbst!

Garbage Collectible Heap

```
Gummibär cubbi;
cubbi = new Gummibär ();

Gummibär sunni =
new Gummibär ();

// Referenzbits kopieren
cubbi = sunni;

if (cubbi == sunni) {
    // gleiche Referenz
}
```



Garbage Collectible Heap

Gummibär cubbi;

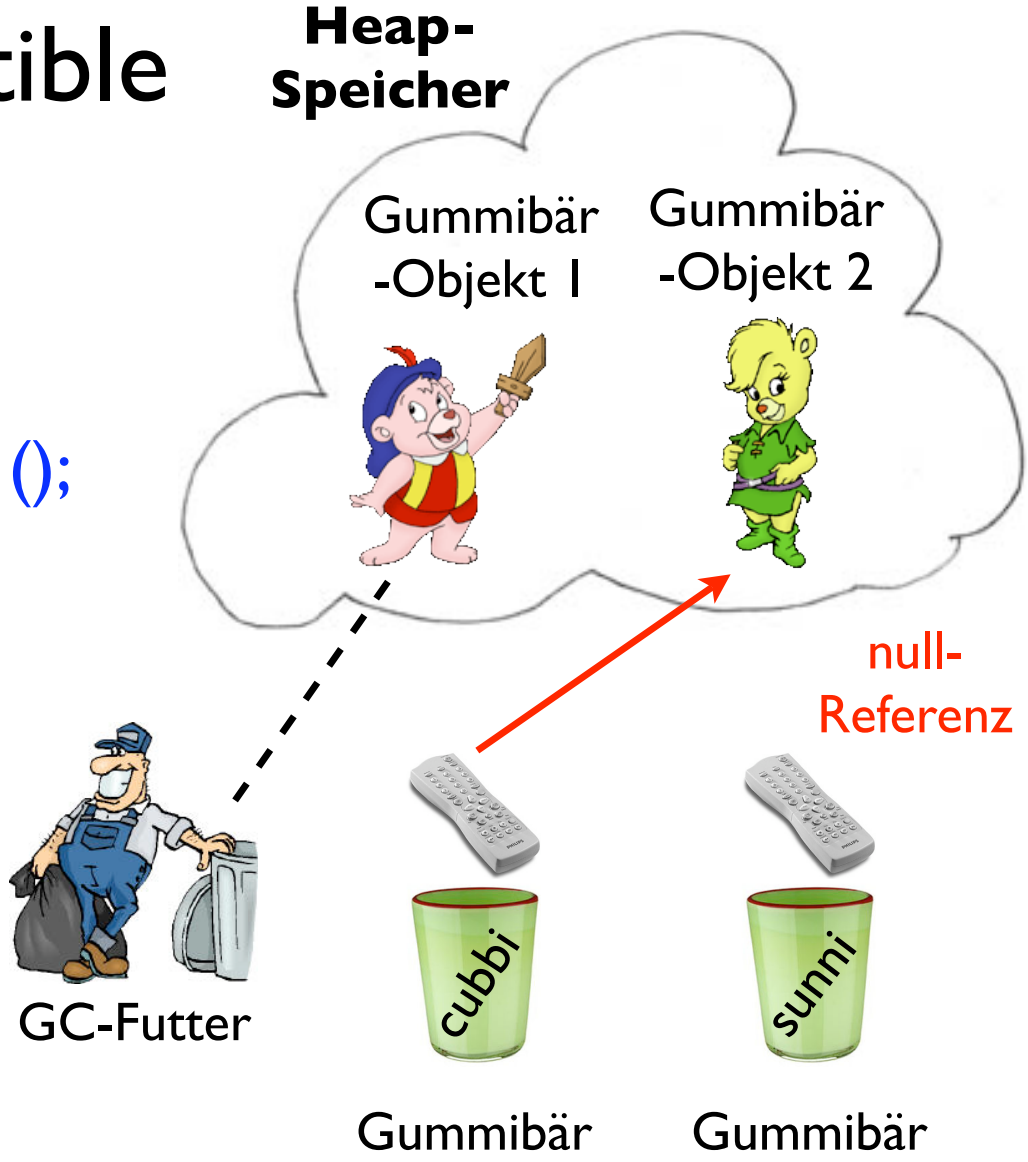
cubbi = new Gummibär ();

Gummibär sunni =
new Gummibär ();

// Referenzbits löschen

sunni = null;

```
if (sunni != null) {
   // Zugriff ok
}
```



Garbage Collectible Heap

Gummibär cubbi;

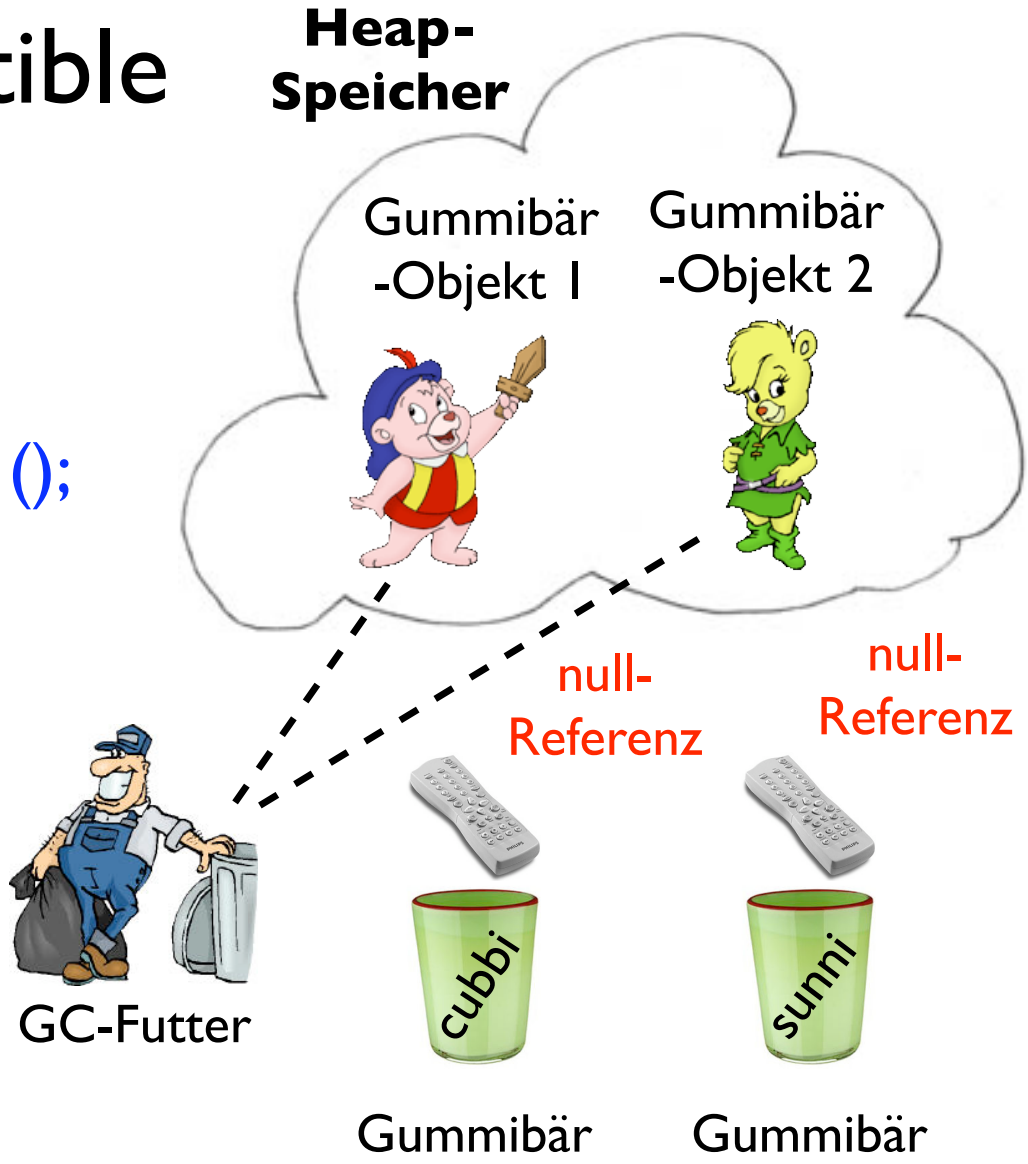
cubbi = new Gummibär ();

Gummibär sunni =
new Gummibär ();

// Referenzbits löschen

sunni = null;

cubbi = null;

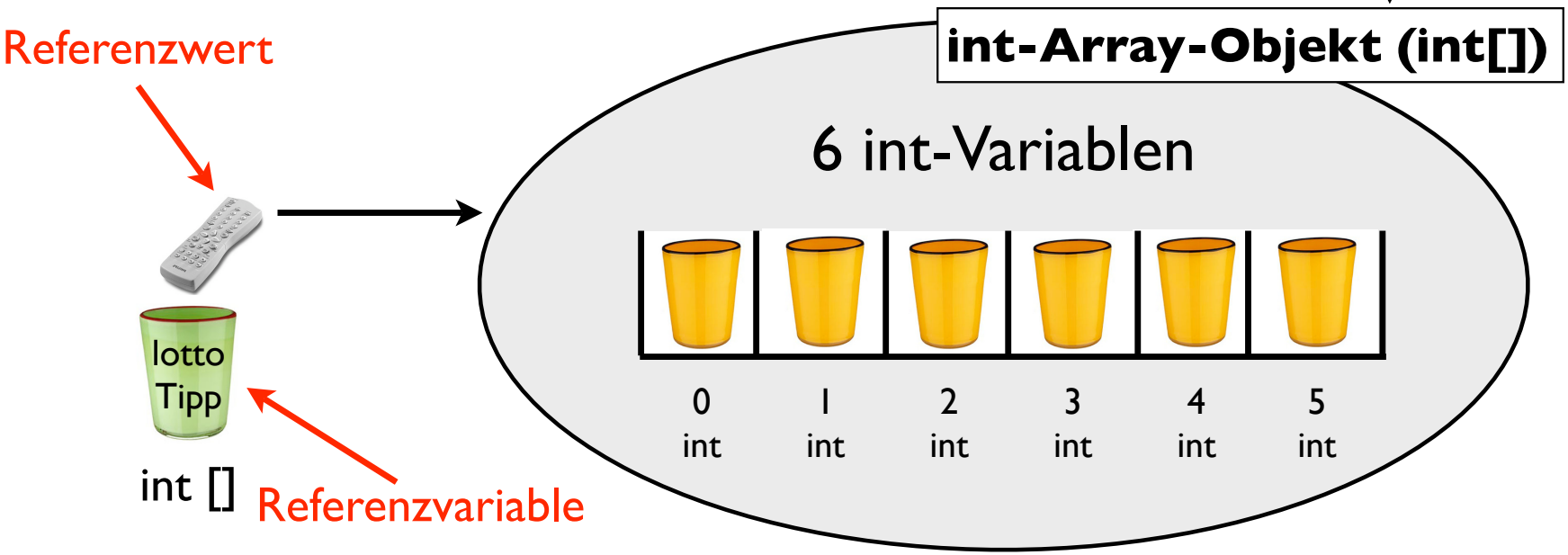


Arrays

speichert nur Eier

```
// int-Array-Variable deklarieren
int[] lottoTipp;

// int-Array-Objekt erzeugen und der Var. zuweisen
lottoTipp = new int [6];
```

speichert nur ints



// Zugriff über Indizes

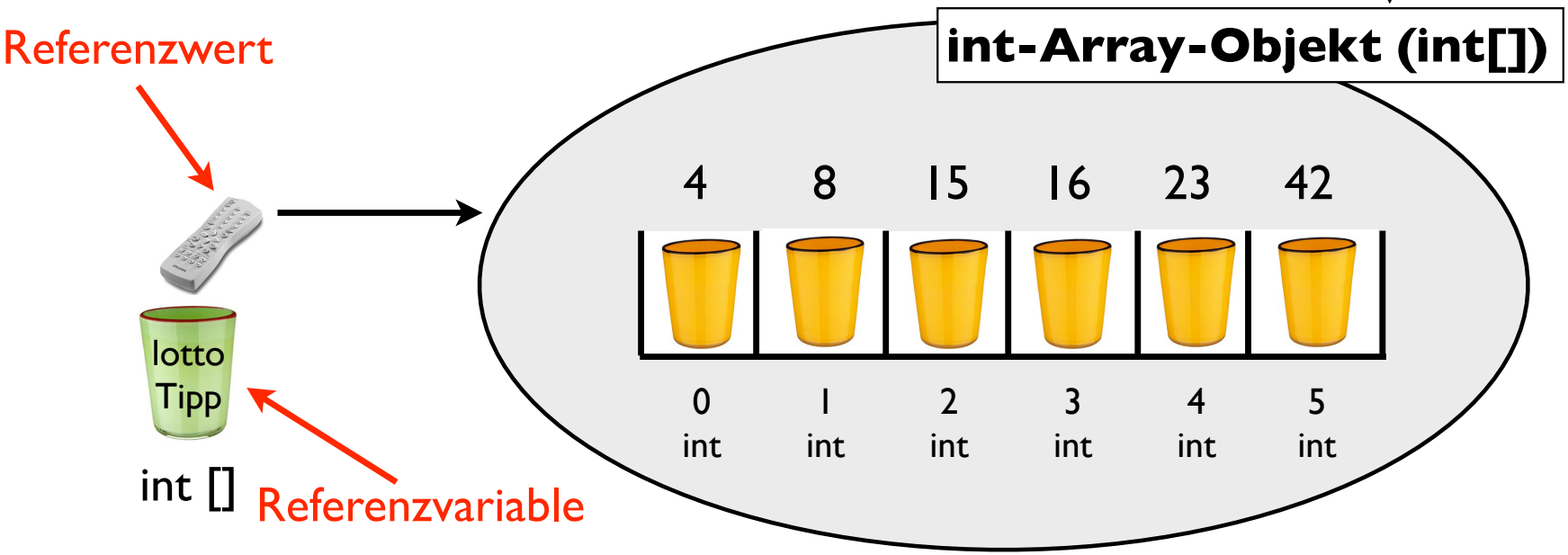
```
lottoTipp[0] = 4;   lottoTipp[1] = 8;
lottoTipp[2] = 15;  lottoTipp[3] = 16;
lottoTipp[4] = 23;  lottoTipp[5] = 42;
```

speichert nur Eier

// deklarieren und initialisieren

```
int[] gewinnZahlen = {4, 8, 15, 16, 23, 42, 49};
```

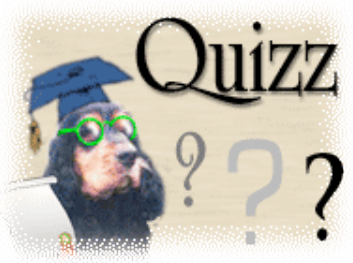
speichert nur int



```
// Länge des Arrays abfragen
while (x < lottoTipp.length) {
      // Gewinnzahlen überprüfen...
}
```

speichert nur Eier

```
float[][] eierGewicht;
eierGewicht = new float[6][5];

eierGewicht.length = ?              // 6
eierGewicht[0].length = ?           // 5
eierGewicht[9].length = ?           // Laufzeitfehler!
eierGewicht[3][1].length = ?        // Compilerfehler!
```

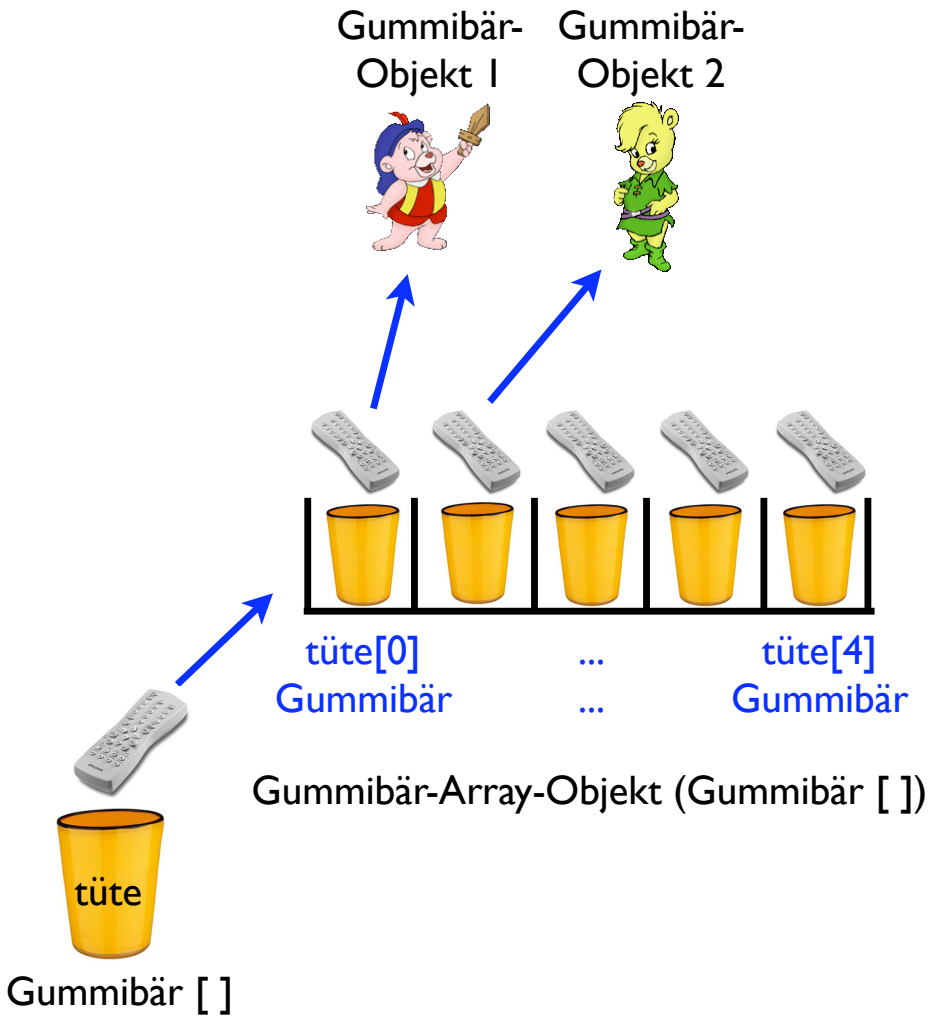


Ein Array voller Gummibären

```
Gummibär[] tüte;
tüte = new Gummibär [5];

// Was fehlt noch?
tüte[0] = new Gummibär ();
tüte[1] = new Gummibär ();
...
```

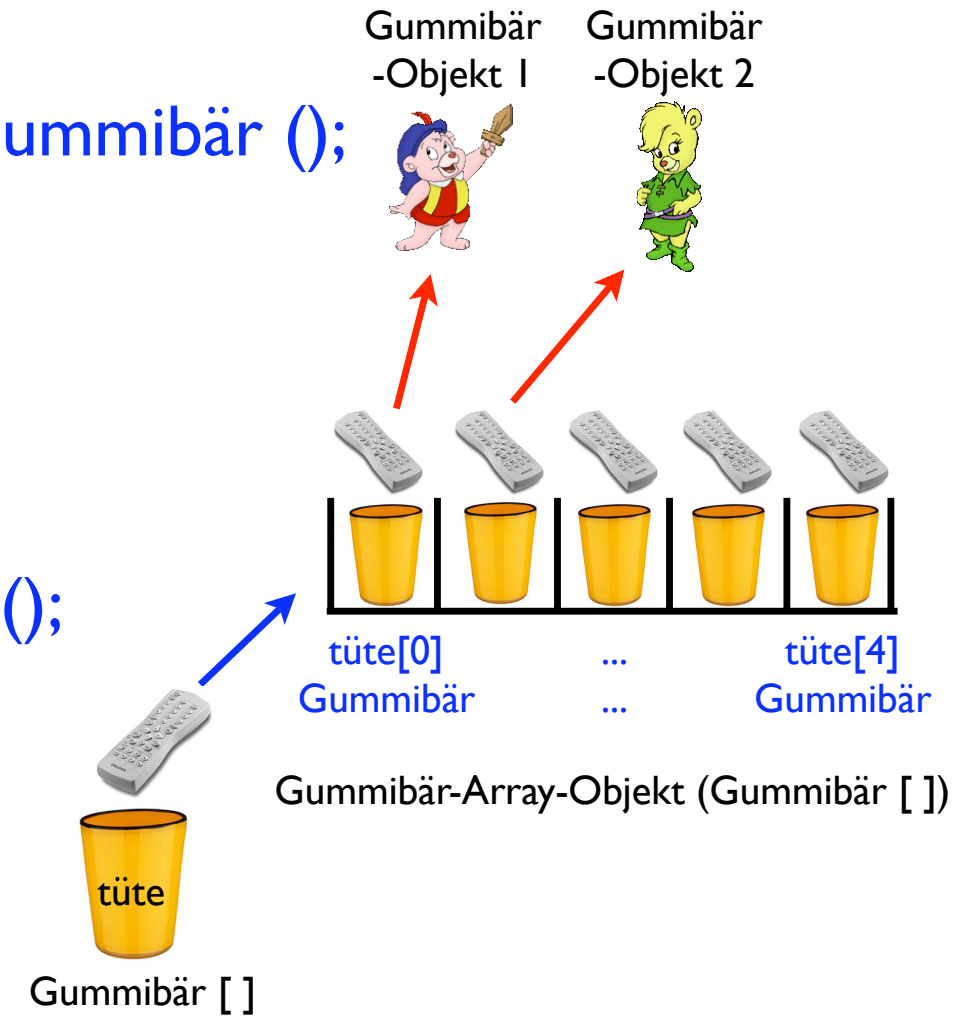
Wir brauchen individuelle Gummibär-Objekte!

Das Array speichert nur die Referenzen auf die Objekte!



Objekte im Array ansprechen

```
// Zugriff bisher
Gummibär cubbi = new Gummibär ();
cubbi.name = "Cubbi";
cubbi.springen ();

// Zugriff aus dem Array
tüte[0] = new Gummibär ();
tüte[0].name = "Cubbi";
tüte[0].springen();
```



```
String titel = new String ("Strings sind Objekte!");
String titel = "Strings sind Objekte!";
```

doppelte Anführungszeichen

```
String leererString = "";

String vorlesung = "Programmierung";
vorlesung = vorlesung + "für Alle";       // konkatenieren
vorlesung = vorlesung.toLowerCase ();

String semester = "WS 06/0" + 7;          // "WS 06/07"
```

```
if (vorlesung == titel) {
    // gleiche Referenzen
}
if (vorlesung.equals (titel)) {
    // gleicher Inhalt der String-Objekte
}

String[] vorlesung = {“Progra”,“4”,“all”};

public static void main (String[] args) {
    System.out.println (args[0]);
}
```

Was wird ausgegeben?

SMS mit Ergebnis an:


```
String[] vorlesung = {“Progra”,“4”,“all”};

String text = vorlesung[0] + vorlesung[1];

System.out.println (“Ich hoere” + text + vorlesung[2]);
```

Methoden & Instanzvariablen

Parameter & Rückgabewerte

Lokale Variablen

Oh nein! Ein Folie ohne Bilder!

Sorry, aber das ist mir viel zu unflexibel!

```
// Zahlen von 1 bis n summieren


    int summe = 0;  int zahl = 1;  int n = 100;

    while (zahl <= n) {
        summe = summe + zahl;
        zahl ++;
    }
```

Argument übergeben...
...Ergebnis zurückgeliefert

Rückgabetyp muss

kompatibel zum

Rückgabewert sein!

```
// Zahlen von 1 bis n summieren
int summiere (int n) {          // Parameter n

    int summe = 0;  int zahl = 1;

    while (zahl <= n) {
        summe = summe + zahl;
        zahl ++;
    }
    return summe;    // Rückgabe von summe
}
```

Muß ich etwas zurückliefern... ich meine, als Gegenleistung für den Parameter?

```
void summiere (int n) {
    int summe = 0; int zahl = 1;
    while (zahl <= n) {
        summe = summe + zahl;
        zahl ++;
    }
}
```

So geht's auch...
das ist aber ... ähm... nutzlos.

Spielen Sie Compiler!


```
int summiere (int n) {
    // Zahlen von 1 bis n summieren
    return summe;
}
```

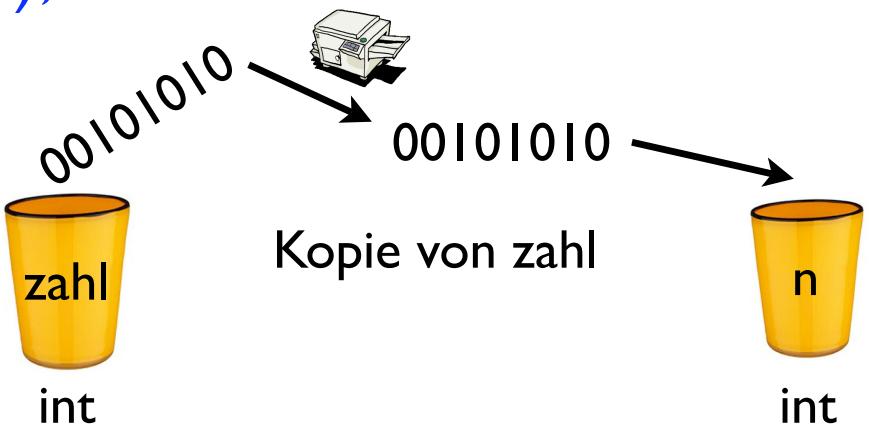
```
int ergebnis;
ergebnis = summiere (25.9);   // falscher Argumenttyp
ergebnis = summiere (25) + ergebnis;   // ok
summiere (ergebnis);          // ok, Ergebnis unwichtig
float sum = summiere (55);    // falscher Typ
long sum = summiere (23);     // ok, Typen kompatibel
```

Pass-by-Value (Pass-by-Copy)

```
int summiere (int n) {
    // Zahlen von 1 bis n summieren
    return summe;
}
```

```
int ergebnis;  int zahl = 42;
ergebnis = summiere (zahl);
```

Bei der Übergabe werden die Bits der Variablen kopiert!



Mehrere Argumente übergeben...

```
// Zahlenintervall summieren
int summiereIntervall (int anfang, int ende) {
    // Summe berechnen
    // Ergebnis zurückliefern
}
```

```
int luftballons = 99;
ergebnis = summiereIntervall (25, luftballons);
ergebnis = summiereIntervall (13, summiere (100));
```

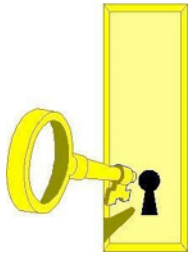
Unglücke heraufbeschwören...

```
class Flugzeug {
  int anzahlMotoren;
  int reichweite;
  ...
}

Flugzeug a380 = new Flugzeug ();
...

// ...irgendwo mitten im Flug...
// ...in der grossen, weiten Java-Welt...
// ...aus einer bösen Klasse heraus...
a380.anzahlMotoren = 0;
```

Hilfe!!!

Was ist da bloß passiert???

Kapselung: Zugriff nur für Befugte!



```
class Flugzeug {
  private int anzahlMotoren;
  ...
  // Setter setzt den Wert der Instanzvariablen
  public void setAnzahlMotoren (int anzahl) {
        if (anzahl >= 1) {
           anzahlMotoren = anzahl;
        }
  }
  // Getter liefert den Wert der Instanzvariablen
  public int getAnzahlMotoren () {
     return anzahlMotoren;
  }
  ...
}
```

Zugriffsmodifier

public:
Zugriff ist für ALLE möglich!

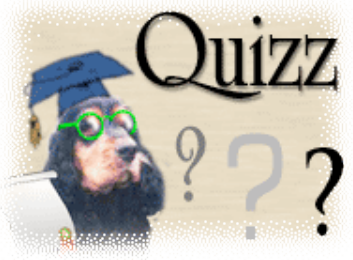
private:
Zugriff nur für Code aus dieser Klasse!

```
class Flugzeug {
   private int anzahlMotoren;

   public void setAnzahlMotoren (int anzahl) {
      if (anzahl >= 1) { anzahlMotoren = anzahl; }
   }

   public int getAnzahlMotoren () {
      return anzahlMotoren;
   }
}
```

```
// irgendwo in der Java-Welt...
Flugzeug a380 = new Flugzeug ();

int anzahl = a380.getAnzahlMotoren ();  // ???
a380.anzahlMotoren = 0;
a380.setAnzahlMotoren (4);
```

Juhuu!
Eine Demo!

Aber vorher kommt ein



Instanzvariablen vs. lokale Variablen

```
class Flugzeug {
   // Instanzvariablen werden mit 0 initialisiert
   private int anzahlMotoren;

   public void setAnzahlMotoren (int anzahl) {
         // lokale Parameter-Variablen werden beim
         // Methodenaufruf initialisiert
   }

   // Wie werden lokale Variablen initialisiert?
   public void beschleunigen () {
         int motorNummer;
         while (motorNummer <= anzahlMotoren) {
               // mehr Sprit für Motor motorNummer
         } } }
```

Compiliert so nicht!

motorNummer müssen Sie vor der Benutzung initialisieren!

Jetzt sind Sie wieder an der Reihe!

Massieren Sie Ihre Neuronen
mit dem Inhalt der **Kapitel 3 + 4**.