Definition einer Klasse mit Attributen

```
public class House { // Def. einer öffentlichen Klasse "House"
  int floors;        // Variable mit dem einfachen Typ "int"
  float height;      // Variable mit dem einfachen Typ "float"
  float length;      // ...
  float width;       // ...
  boolean hasCellar; // Variable mit dem einfachen Typ
   "boolean"
  short nOfLifts;    // Variable mit dem einfachen Typ "short"
  float roomSizes[]; // Referenz auf ein Array of "float"
  Person owner;      // Ref. auf ein Objekt vom Typ "Person"
  Person renters[];  // Referenz auf ein Array of "Person"'s
} // Ende der Klassendefinition
```

Referenzvariablen

```
Person unknown, myself = new Person ();
unknown = null;          // referenziert kein Objekt
myself.name = "Martin";
unknown = myself;        // referenziert nun das gleiche Objekt
                         // wie "myself" (keine Kopie!)
writeLn (unknown.name); // Ausgabe: "Martin"
unknown.name = "Franz";
writeLn (unknown.name); // Ausgabe: "Franz" (wie erwartet)
writeLn (myself.name);  // Ausgabe: "Franz" (Seiteneffekt!!!)
myself = new Person();
myself.name = "Martin";
writeLn (myself.name);  // Ausgabe: "Martin"
unknown.name = "Max";
writeLn (unknown.name); // Ausgabe: "Max"
writeLn (myself.name);  // Ausgabe: "Martin" (kein Seiteneff.)
unknown = null;
writeLn (unknown.name); // Fehler!!! NullPointerException
```

Methodendeklaration

```
public class House {
  private short nOfLifts; // Nicht nach aussen sichtbar
  ...
  //Methode ohne Eingabeparameter und mit Rückgabewert (short)
  short getNOfLifts () {
    return nOfLifts; // „Gib nOfLifts zurueck“
  }
  // Methode mit einem Eingabeparameter (short) und
  // ohne Rückgabewert (void)
  void setNOfLifts (short newValue) {
    if (newValue < 0) { // Schutz vor ungültigen Werten
      newValue = 0;     // Was tun bei ungültigem Wert?
    }
    nOfLifts = newValue;
  }
}
```

Methodenaufruf

```
House myHouse1 = new House();

...

short x = myHouse1.getNOfLifts (); // Lesender Zugriff
writeLn (myHouse1.getNOfLifts ());
myHouse1.setNOfLifts ((short)2);   // Schreibender Zugriff
myHouse1.setNOfLifts ((short)2 * x);

myHouse1.setNOfLifts ((short)-3);  // abgefangener Fehlerfall
writeLn (myHouse1.getNOfLifts ()); // Ausgabe: "0"
```

Komplexere Methoden

```
...
   float calcBase () { // Berechnung der Grundfläche
        return length * width; // Länge mal Breite
   }
   float calcVolume () { // Berechnung des Volumens
        return calcBase () * height; // Grundfläche mal Höhe
   }
   // Berechnung der Gesamtsumme
   float calcRentSum (float squarePrice) {
        float sum = 0.0f;
        for (int i = 0; i < roomSizes.length; i ++) {
                sum += squarePrice * roomSizes [i];
        }
        return sum;
   }
...
```

Strings

```
String name = null;
                // Variable ohne Referenz auf Zeichenkette

name = "Otto"; // Neuer String mit Zeichenkette "Otto"
String name2 = name;
writeLn (name2 == name);  // => true, weil dasselbe Objekt

name2 = new String ("Otto"); // Neuer, initialisierter String
writeLn (name2 == name);  // => false, weil nicht dasselbe
                          // Objekt, nur gleiche Zeichenkette

writeLn (name2.equals (name));// => true, weil Zeichenkette
                              // verglichen wird

writeLn (name2.length ()); // => 4, "Otto" hat 4 Buchstaben
```