Klassen in Kotlin

```
fun main() {

  val welt = Welt(10, 10, 6)
  val robi = Roboter(welt)
  var schritte = 0

  fun zurWandGehen() {
    while (!robi.IstWand()) {
        robi.Schritt()
        schritte = schritte + 1
    }
  }
  // und so weiter ...
}
```

```
class Baum() {

  var standort = ""
  var art = ""
  var stammdurchmesser = 0
  var aktualisierungNoetig = false

  fun zuwachsRegistrieren(zuwachs: Int) {
    stammdurchmesser = stammdurchmesser + zuwachs
  }

  // und so weiter ...

}
```

Die Struktur der Klasse ist mit der der ‚äußeren' Funktion im linken Beispiel vergleichbar: Erst werden Variablen deklariert, um darin Daten zu speichern – das sind die Attribute – dann folgen Funktionen, die mit den Daten arbeiten – das sind die Methoden. Allerdings wird eine Funktion, wenn sie aufgerufen wird, von Anfang bis Ende abgearbeitet und dann werden die darin deklarierten Variablen wieder gelöscht. Lasse ich aber eine Klasse ein Objekt erzeugen und speichere dies unter einem entsprechenden Bezeichner (auch dafür werden Variablen verwendet), dann bleiben die Attribute mit ihren Werten erhalten.

Im Beispiel sind die anfänglichen Attributwerte eines Baum-Objekts noch nicht sinnvoll; sie müssen, wenn das Objekt erzeugt wurde, noch richtig gesetzt werden (siehe rechts). Das geht auch besser: Bei der Erzeugung des Welt-Objekts im Beispiel links oben werden ja auch gleich Angaben zur Größe der Welt als aktuelle Parameter übergeben, die dann sicherlich in Attributen gespeichert werden. Damit das in der Klasse Baum möglich wird, müssen in die Parameterliste entsprechende Parameter aufgenommen werden, und diese müssen mit dem Schlüsselwort **val** oder **var** als Attribute gekennzeichnet werden:

```
val b1 = Baum()
b1.standort = "Haupteingang Friedhof rechts"
b1.art = "Acer Platanoides Deborah"
b1.stammdurchmesser = 33
```

```
class Baum(val standort: String, val art: String, var stammdurchmesser: Int) {

  var aktualisierungNoetig = false

  fun zuwachsRegistrieren(zuwachs: Int) {
    // und so weiter ...
  }
}
```

So ist dafür gesorgt, dass das entsprechende Baum-Objekt gleich nach der Erzeugung wirklich „fertig" ist:

```
val b1 = Baum("Haupteingang Friedhof rechts", "Acer Platanoides Deborah", 33)
```

Weil mithilfe der Festlegungen nach dem Schlüsselwort **class** ein Objekt der Klasse „fertig konstruiert" werden kann, nennt man sie den Konstruktor der Klasse.