Klassen und Objekte

- Motivation
- Bankkonten und ihre Implementierung
- Begriffe
- Zustände: Aufzug-Beispiel

Zustandslose Methoden

Fast alle bisherigen Methoden sind *nur* abhängig von ihren Parametern oder von einer Eingabe; Beispiele

- int readInt()
- boolean isPerfectNumber(int n)
- assertEquals(int expectedValue, int actualValue)
- boolean success(String topLeft, String topCenter, ...)
- druckeEtikett(String farbe, boolean qrVorhanden, int breite)

Ein Beispiel, in dem das anders ist

- Wie ist das mit einem Bankkonto?
- Welche Methoden lassen sich auf ein Bankkonto anwenden?
- Von was hängen diese Methoden ab?
  Was brauchen wir also?

Beispiel (Fortsetzung)

- Wie ist das mit einer Bank?
- Welche Methoden lassen sich auf eine Bank anwenden?
- Von was hängen diese Methoden ab?
  Was brauchen wir also?

Viele Konten in der Bank

- Programmiert haben wir Klassen

```
class Konto { ... }
class Bank { ... }
```

- Eine Bank hat *viele* Konten;
  diese stellen wir dar als *Objekte* nach dem *Muster ihrer Klasse*
- Wie bei Arrays erzeugen wir ein *neues* Konto mit „new“:

```
new Konto()
```

HOCHSCHULE DER WIRTSCHAFT FÜR MANAGEMENT
UNIVERSITY OF APPLIED MANAGEMENT STUDIES

Erzeugen von Konten

- Wie bei Arrays erzeugen wir ein *neues* Konto mit „new“:

  new Konto()

- Wie bekommen wir z.B. den Inhaber und die Kontonummer in das neue Konto-Objekt?
  → als Parameter

  new Konto(“Meier, Kurt“, 1234567890)

Erzeugen von Konten

- Wie bekommen wir z.B. den Inhaber und die Kontonummer in das neue Konto-Objekt?
  → als Parameter

```
new Konto( “Meier, Kurt“, 1234567890 )
```

- Jetzt müssen wir dafür sorgen, dass die Information im Konto abgelegt wird
  → Wir benötigen einen *Konstruktor* für die Klasse „Konto“

```
Konto( String kontoInhaber, int kontoNummer ) {
        inhaber = kontoInhaber;
        nummer = kontoNummer;
}
```

- Genau dieser Konstruktor wird mit dem „new“ oben aufgerufen und liefert das neue Objekt als Ergebnis

Ein Hauptprogramm

- Bank erzeugen

- Konto für Hr. Meier anlegen
- Ergebnis anzeigen lassen

- Konto für Hr. Schmidt anlegen
- Ergebnis anzeigen lassen

- Konto von Hr. Meier nochmal kontrollieren

Nächste Schritte

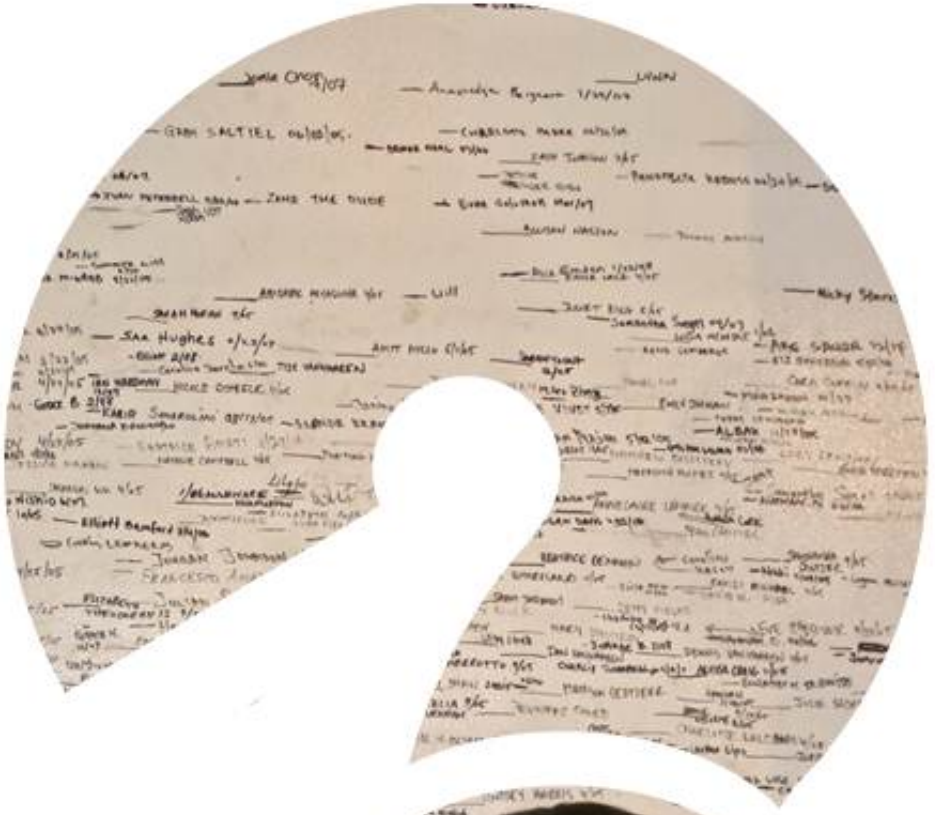
- Herr Müller will mittels Kontonummer auf sein Konto zugreifen:
  Kennt er die / Woher kennt er die?

- Seinen Kontostand abfragen

- Einzahlen auf sein Konto

Erkenntnis

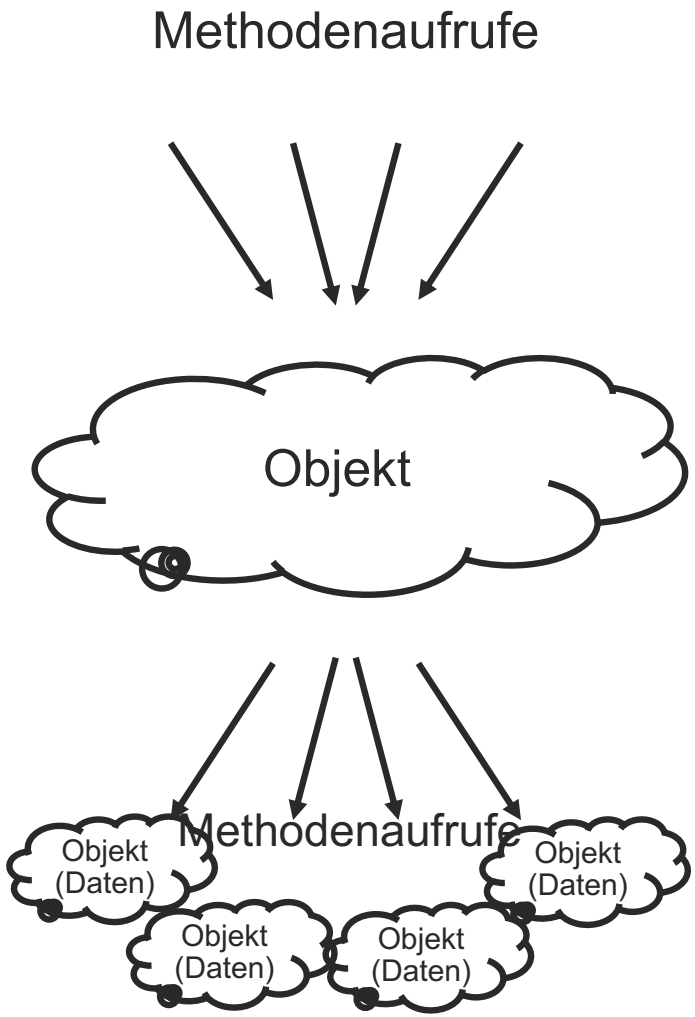
- Mit *Attributen* kann jedes Objekt seinen eigenen individuellen Zustand zwischen zwei Methodenaufrufen erhalten
- Dadurch wird es möglich, zustandsabhängige Methoden zu erstellen



F R A G E N

Konzept der objektorientierten Programmierung

- *Daten* (Objekte) stehen *im Mittelpunkt*
- *Nur* mittels *Methodenaufrufen* können wir mit ihnen arbeiten, z.B. etwas abfragen oder eine Zustandsänderung bewirken



Objekte haben Zustände 1/2

Ein Aufzug als Objekt:

Was kann man mit einem Aufzug machen?
Welche Methoden hat er/braucht er?

Welche Zustände kann er haben?

Wie kann seine Schnittstelle aussehen?

Perspektive des *Benutzers*

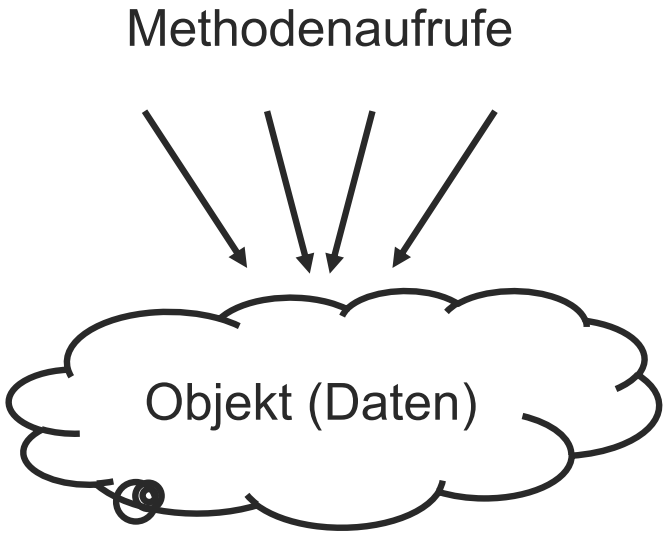
Wir sagen:
Aufzug, fahre in den 8. Stock!

Wir ziehen den Aufzug ***nicht*** selbst!
Nur der Aufzug selbst kann seinen Zustand ändern!

Zustandsänderung, Umgang mit Objekten

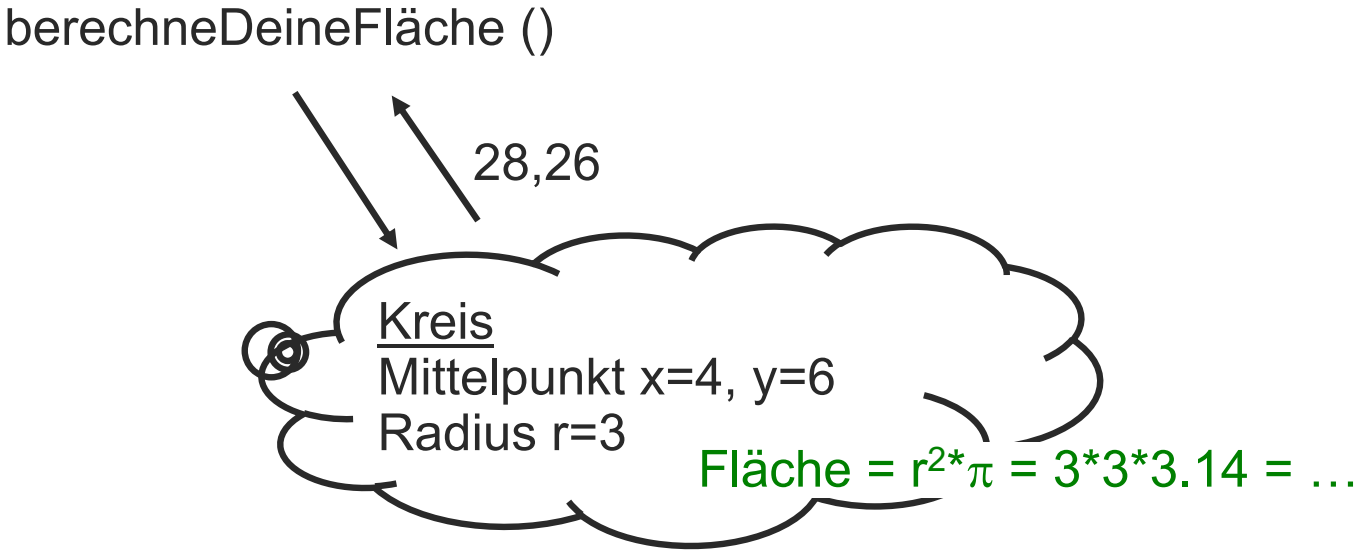
Daten (Objekte) stehen *im Mittelpunkt*,
Methodenaufrufe bewirken eine **Zustandsänderung**

Lokalitätsprinzip:
Alles was mit den Daten arbeitet (und nur das), wird zusammen (in einer Klasse) implementiert, z.B. der Aufzug inkl. Zustandsinfo und Methoden



In der Programmiersprache: Beispiel für Objekte 1/2

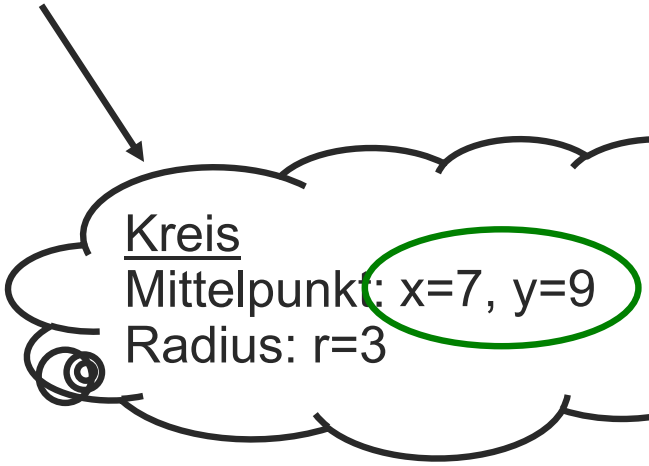
Daten (Objekte) stehen im Mittelpunkt, man kann ihren Zustand abfragen / sie mit ihren Zustandsdaten arbeiten lassen



In der Programmiersprache: Beispiel für Objekte 2/2

Daten (Objekte) stehen im Mittelpunkt,
Methodenaufrufe bewirken eine Zustandsänderung

verschiebeDich (3, 3)



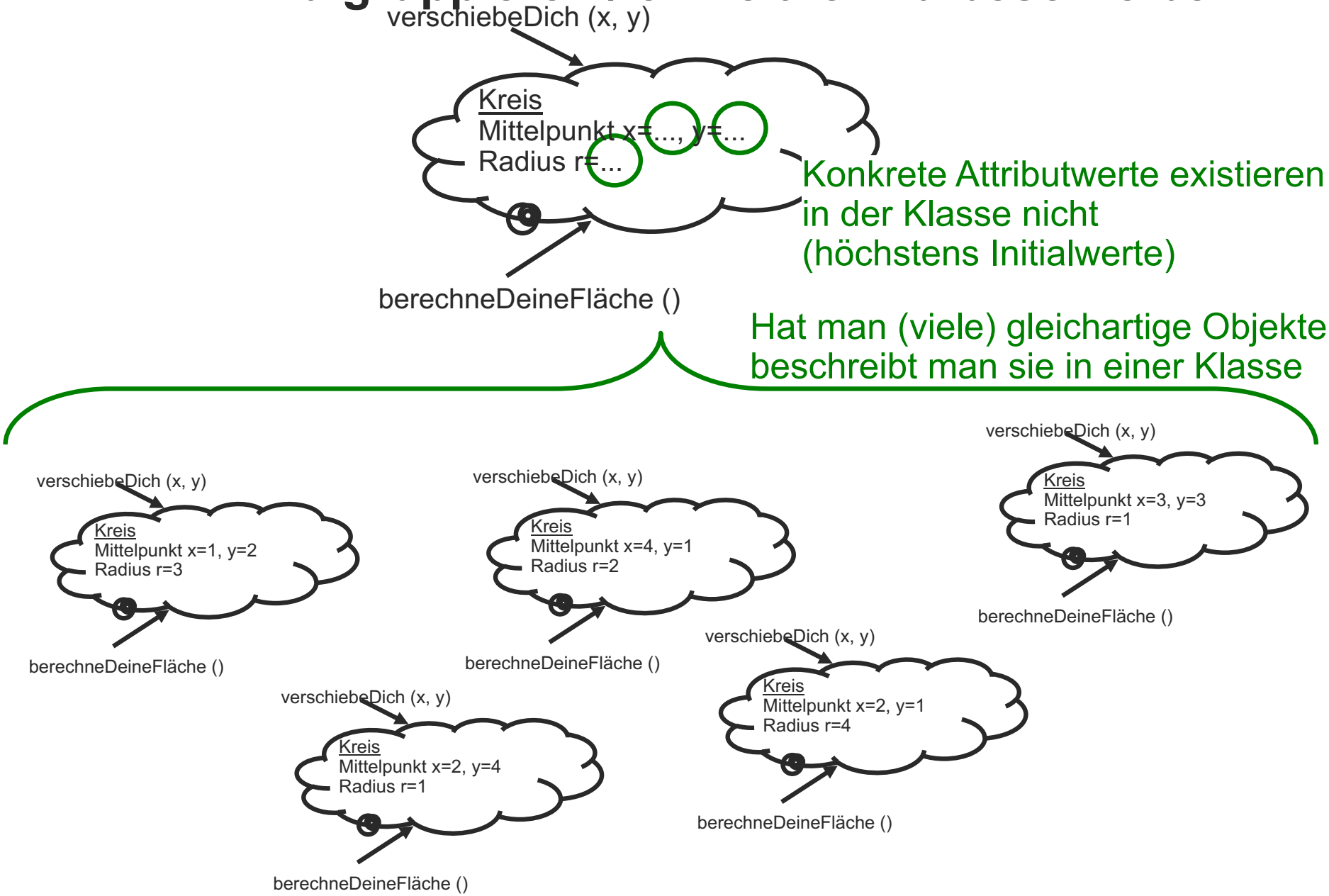
Der Objektzustand wird geändert, aber ***nur vom Objekt selbst*** *(= von Methoden der Klasse des betroffenen Objekts)*!

Wir wenden uns an den Kreis, in Java-Schreibweise:

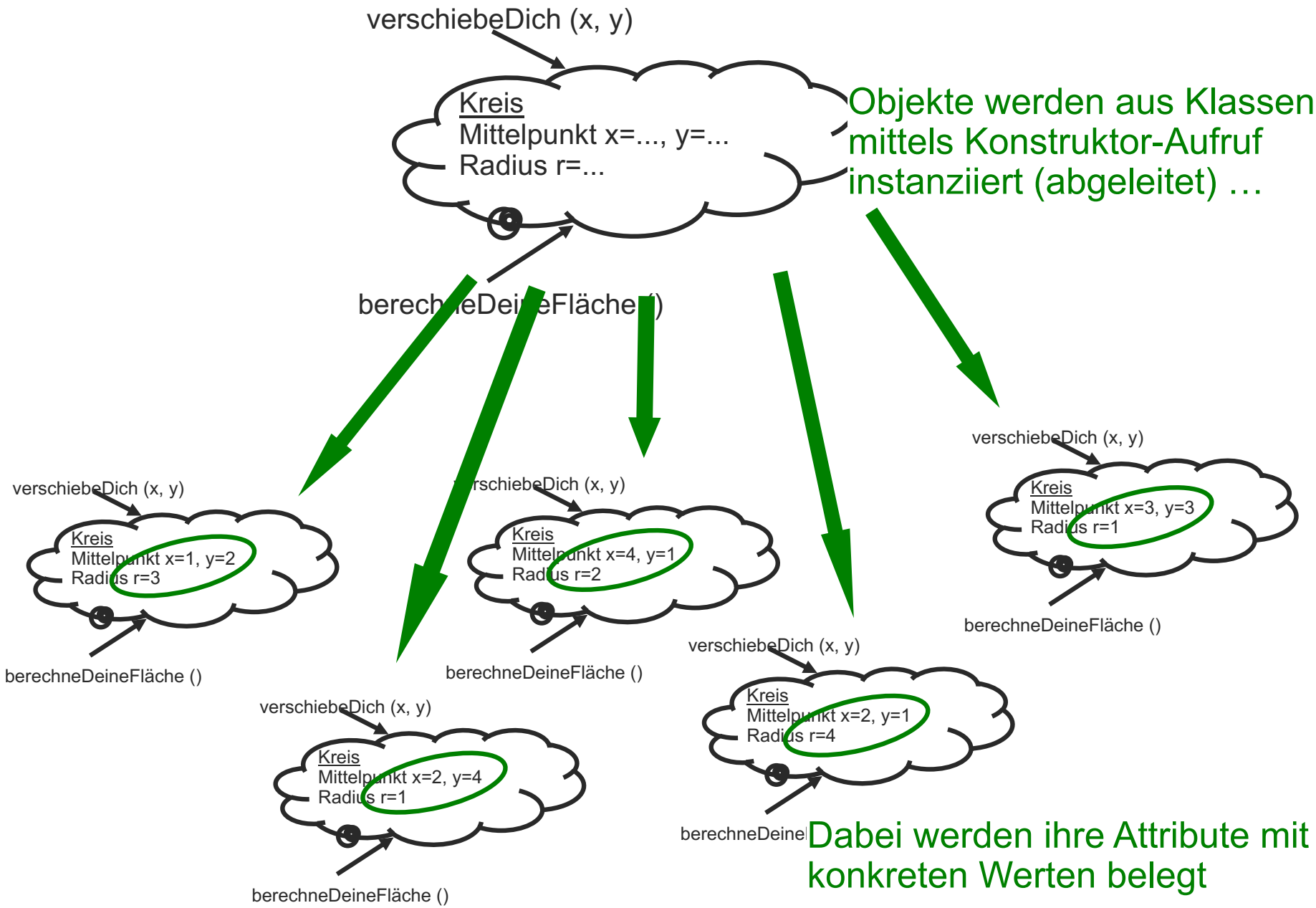
```
Kreis k = new Kreis();
k.verschiebeDich( 3, 3 );
```

Daten und die zugehörigen *Methoden*
werden in einer Klasse *zusammengefasst*,
nur die Methoden einer Klasse selbst
greifen auf ihre Attribute zu (können zugreifen)

Man benutzt Klassen, um gleichartige Objekte zu gruppieren / einheitlich zu beschreiben



Zur Laufzeit eines Programms existieren nicht die Klassen (d.h. die Java-Dateien), sondern nur deren Objekte (Instanzen)



Klassen und Objekte

- Eine **Klasse** *beschreibt* eine (möglicherweise sehr große) Menge *gleichartiger Objekte* mit individuellenZuständen (= individuelle Attributwerte: jeder Kreis liegt an einem anderen Platz, hat eine andere Größe)

- Jedes **Objekt** ist eine eindeutige Instanz seiner Klasse
  Es *kennt seine Klasse*

F R A G E N