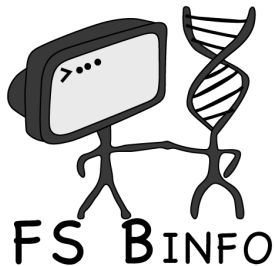


# Objektorientierte Programmierung



Rebecca Kaßner

Fachschaft Bioinformatik

17. Januar 2012

# Zeitplan

---

- 25.10 Einführung in Linux und die Shell
- 08.11 Arbeiten mit Dateien auf der Shell
- 15.11 Shell III und Editor: vim
- 22.11 Einführung in Java und Eclipse
- 29.11 Konditionen, Schleifen und Arrays
- 06.12 Rekursion und Iteration
- 13.12 File - Input/Output
- 20.12 Weihnachtsübung
- 10.01 Debuggen mit Eclipse
- 17.01 **Objektorientierte Programmierung**
- 24.01 Vererbung und Tipps
- 30.01 Quiz, Fragen, Evaluation

Webseite: <http://www.bioinformatik-muenchen.com/bioinfocom/informatik-tutorium>

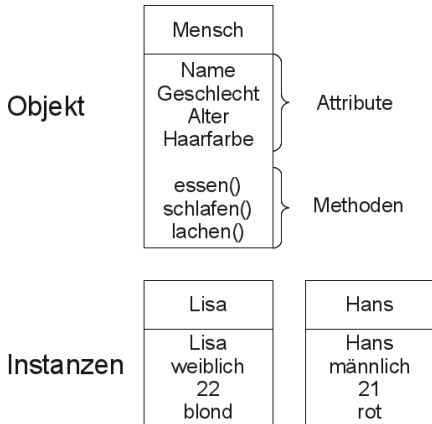
# Was ist Objektorientierte Programmierung (OOP)?

---

- Der Mensch nimmt die Welt als eine Sammlung von Objekten wahr
- Logische Einheiten in Objekte verpacken um die Realität besser abbilden zu können
- Die vier Grundsäulen:
  - Abstraktion
  - Kapselung
  - Vererbung
  - Polymorphie

# Objekte, Methoden, Instanzen, Attribute ... ??

- Objekte werden mit Attributen und Methoden definiert
- Es können mehrere Instanzen von einem Objekt erstellt werden
- Die Instanzen unterscheiden sich durch ihre Attribute, haben aber alle dieselben Methoden



# Was ist eine Klasse/Objekt?

---

- Werden mit `class` gekennzeichnet
- Klassenname wird meist groß geschrieben
- Können Attribute und Methoden haben (Zugriff mit `.`)
- Instanzen von Objekten werden mit `new` erzeugt

```
class Counter {  
    int counter = 0; /* Attribut */  
    int increment() { /* Methode */  
        counter++;  
        return counter;  
    }  
    public static void main (String[] args) {  
        Counter c = new Counter ();  
        System.out.println(c.increment ());  
    }  
}
```



# Konstruktoren

---

- Werden bei der Konstruktion eines Objekts mit `new` aufgerufen
- Besondere Signatur: kein Rückgabewert, selber Name wie Klasse
- Mit `this` greift man auf Attribute und Methoden der eigenen Klasse zu
- Hilft wenn lokale Variablen denselben Namen wie Objektvariablen haben

```
class Counter {  
    int counter;  
  
    public Counter(int initial_value) {  
        this.counter = initial_value;  
    }  
}
```



# Übung 1

---

1. Erstelle eine Klasse Kreis mit den Attributen  $x$ ,  $y$  (Lage des Mittelpunktes in einem Koordinatensystem) und Durchmesser
2. Der Konstruktor soll alle 3 Parameter erwarten
3. Implementiere die Methoden `berechneOberflaeche` und `berechneVolumen` die den Umfang bzw. die Fläche zurückliefern sollen

## TIPPS:

- $\pi$ : `Math.PI`
- Quadrieren: `Math.pow(basis, exponent)`
- Umfang:  $\pi * d$
- Volumen:  $\pi * r^2$ )

## Statische Methoden und Attribute

---

- Methoden und Attribute können als `static` deklariert werden
- Im Gegensatz zu Objektvariablen gelten statische Variablen für alle Instanzen der Klasse gleichermaßen
- Statische Methoden können auch aufgerufen werden ohne vorher eine Instanz des Objekts zu erzeugen (siehe `main`-Methode)

```
class Counter {  
    static int counter;  
    static int increment() {  
        counter++;  
        return counter;  
    }  
    public static void main (String[] args) {  
        System.out.println(Counter.increment ());  
    }  
}
```



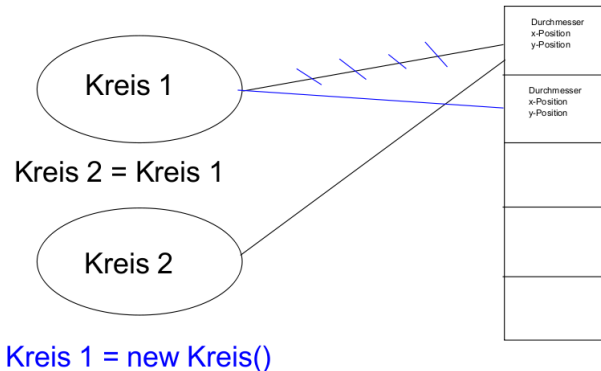
## Übung 2

---

1. Erstelle ein statisches Attribut "anzahl" in der Klasse Kreis, die angibt wie viele Instanzen der Klasse Kreis erzeugt wurden

# Referenzierung

- Objekte sind nur Referenzen auf Speicherpositionen
- Veränderung betrifft jede Referenz auf Speicherposition
- Primitive Datentypen werden direkt verändert



# Übungen für Daheim

---

1. Erstelle die Klassen Trapez, Rechteck und Quadrat und implementiere die Methoden berechneOberflaeche und berechneVolumen wie in Übung 1
2. In welcher Beziehung stehen die Objekte Trapez, Rechteck und Quadrat?