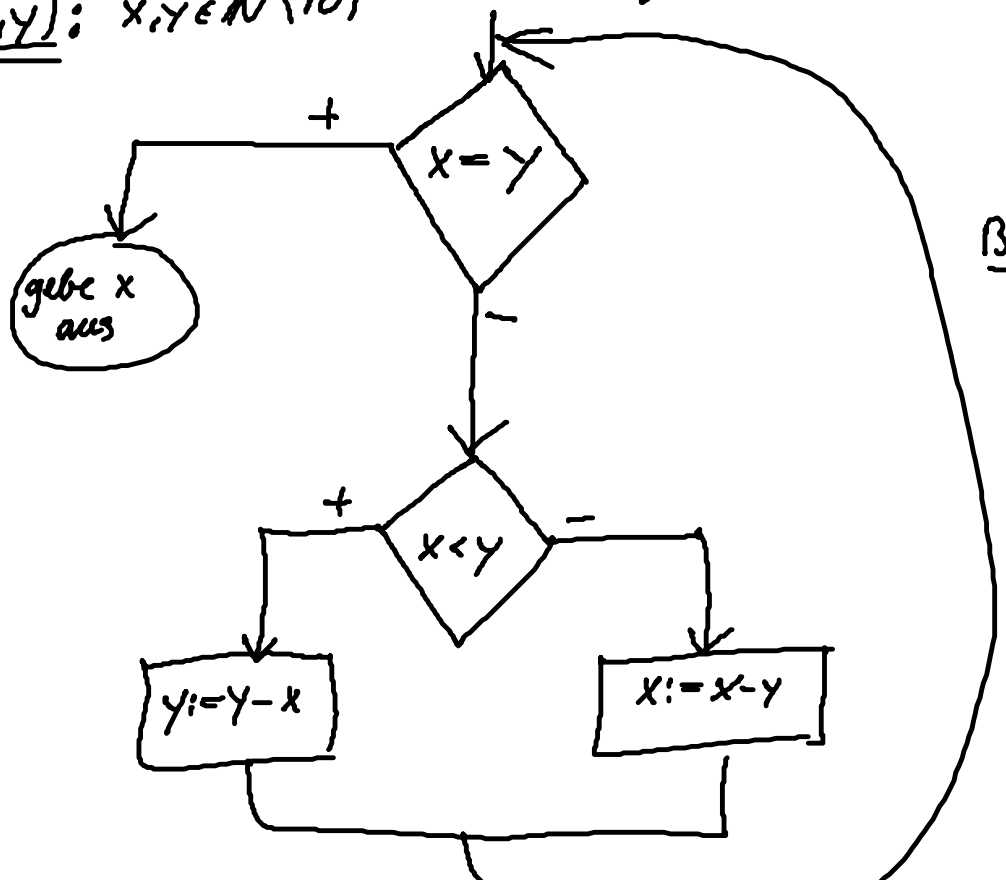
- HA neu von der Di-VL im MA0001
- Evaluation der Fib.-OA aus OA2
  war nicht ganz korrekt.
  Fehler behoben.
  In Einzelfällen haben Studierende ohne korrekten Code volle Punktzahl erhalten.
  Bitte nochmals auswerten.
- Neues Tutorium bei Daniel Kuske Do 10-12

Erinnerung: Für $a > b \geq 1$, $a, b \in \mathbb{N}$ gilt

$$ggT(a,b) = ggT(a-b, b)$$

Damit erhalten wir verbesserten Algorithmus (im Flussdiagramm):

ggT(x,y): $x, y \in \mathbb{N} \setminus \{0\}$



Beispiel:

ggT(28, 12)

16, 12

4, 12

4, 8

4, 4

Programm dazu auf Homepage!

```
Terminal — bash — 80×24
dhcp-15-219:GGT peis$ javac GGT.java
dhcp-15-219:GGT peis$ java GGT
Geben sie eine natürliche Zahl x > 0 ein:
1246
Geben sie eine natürliche Zahl y > 0 ein:
248
Der ggT von 1246 und 248 ist 2.
dhcp-15-219:GGT peis$
```

Weitere Beschleunigung des Algorithmus

Lemma impliziert Folgerung: Für $a > b \geq 1$, $a, b \in \mathbb{N}$ gilt

$$ggT(a,b) = \begin{cases} b & \text{falls } a \bmod b = 0 \\ ggT(a \bmod b, b) & \text{falls } a \bmod b \neq 0 \end{cases}$$

⇝ Euklidischer Algorithmus

Ausdrücke in Java (noch einmal etwas genauer):

bisher: Ausdruck berechnet Wert und hat Typ.

darüber hinaus: Ausdrücke haben Effekte (nur manchmal sichtbar)

`a = b;`

Zuweisung, ist in Java ein Ausdruck
wird durch `;` zu einer Anweisung

Wert ≙ Wert von b
Typ ≙ Typ von a
Effekt ≙ Veränderung von a

```
int a = 2;
int b = 3;
a = b++;
```

`b++` — Ausdruck

Wert ≙ Wert von b
Effekt ≙ b wird um eins erhöht

hinter Anweisung `a = b++;`

a	3
b	4

Falls statt `a = b++;` stünde

`a = ++b;` | a [4] | b [4]

Wert von a ist Wert von b nach Erhöhung.

Seiteneffekt: b wird um eins erhöht.

Merke: a++ ≙ erst benutzen (Wert), dann erhöhen
++a ≙ erst erhöhen, dann benutzen.

analog für "--"

Wichtiges Prinzip (aus C-Programmierung):

Ein Ausdruck wird durch Anhängen eines ";" zu einer Anweisung, der Wert des Ausdrucks wird dabei unterdrückt.

`a++;` } In beiden Fällen wird a um eins erhöht,
`++a;` } der Wert wird nicht genutzt.

(Überlauf (Erinnerung):
`byte b = 127;` b [128] byte hat Wertbereich -128,...,127
`b++;`)

Vorsicht:

a [1]
b [5]
c [2]
vorher

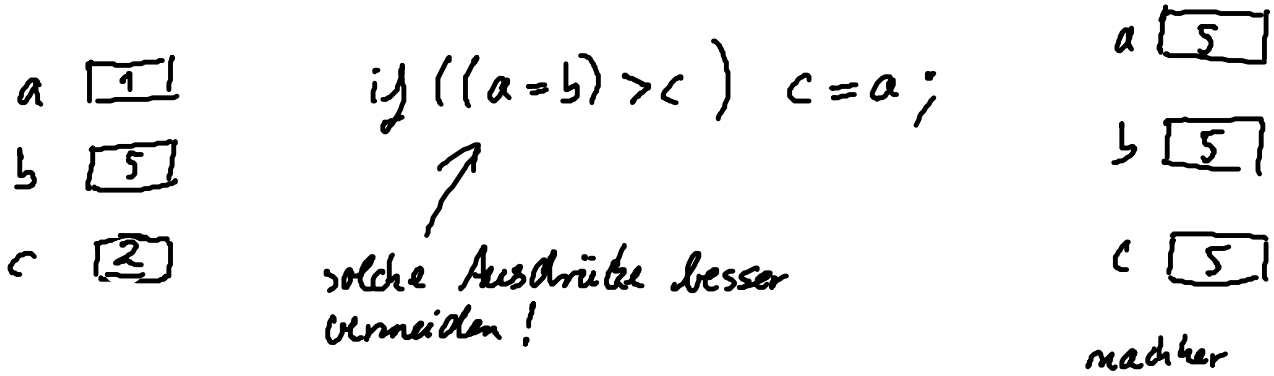
`c += b = a;`
Zuweisungsausdruck mit Wert a und Effekt für b
Zuweisung

a [1]
b [1]
c [3]
nachher

Veränderung des Wertes von b nennt man auch Seiteneffekt!

deutlicher: `b = a;` `c += a;` leistet dasselbe

Weiteres Beispiel:



Klassen: Eine Einführung.

Klassen fassen Objekte mit gleichartigen Eigenschaften zusammen

Bei Konstruktion immer überlegen:

- Was sollen die Objekte?
- Was kennzeichnet sie, d.h., wie drücke ich das mit Variablen aus?
- Was sollen sie können, d.h. welche Methoden brauche ich?

Klassen dienen (wie Methoden) der Strukturierung und Wiederverwendbarkeit.

Es gibt umfangreiche Java Klassenbibliothek (→ Übung letzte Woche)

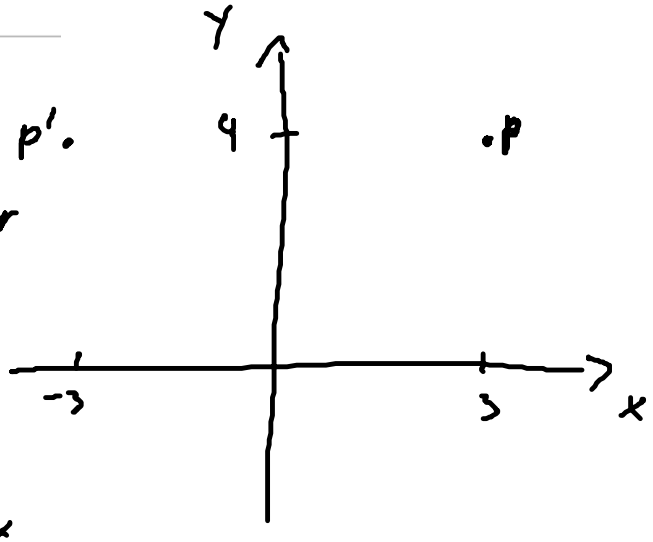
Man kann eigene Klassen schreiben und damit Typen für jeden beliebigen Zustand selbst erzeugen und verwenden und anderen zur Verfügung stellen.

Beispiel: Klasse, deren Objekte Punkte in der Ebene repräsentieren

```java
public class Point {

	// object variables

	private double x;
	private double y;

	// constructors

	public Point() { // default constructor
		this.x = 0;
		this.y = 0;
	}

	public Point(double x, double y) { // standard constructor
		this.x = x;
		this.y = y;
	}

	// set methods

	public void setX(double x) {
		this.x = x;
	}

	public void setY(double y) {
		this.y = y;
	}

	// get methods

	public double getX() {
		return this.x;
	}

	public double getY() {
		return this.y;
	}

	// other methods

	// mirror point at y-axis

	public void mirrorY() {
		this.x = -this.x;
	}

	// string representation

	public String toString() {
		return "(" + this.x + "," + this.y + ")";
	}
}
```

erwartet keine Parameter

Implizite Referenz auf das zu konstruierende Objekt. (neu im Rumpf der Klasse verwendbar)




```
Point p = new Point(3,4);
double a = p.getX();
```

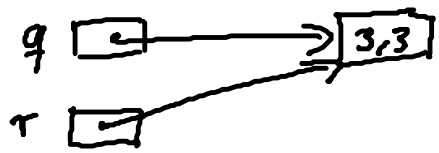
a 3.0

```
Point p' = p.mirrorY();
```

jede Klasse sollte toString() implementieren

erzeugt String (x,y) mit Werten für x und y.

```
Point q = new Point(3,3);
Point r = q;
```



Weitere nützliche Standardmethode ist clone()

erzeugt Kopie des Objekts

```
public Point clone(){
    return new Point ( this.x, this.y );
}
```

Benutzung der Klasse Point:
hier mit Klasse TestPoint

```
public class TestPoint {

        public static void main(String[] args) {


            Point p = new Point(3,4);
            Point q = new Point(3,7);

            System.out.println(" Point p is " + p.toString());
            System.out.println(" Point q is " + q.toString());

        }


}
```

Shell

```
> java TestPoint
Point p is (3,4)
Point q is (3,7)
```

Klasse Point und TestPoint liegen im demselben Directory

Alle Klassen im demselben Directory können die public-Methoden aus anderen Klassen des Directories benutzen.
(nicht die private-Methoden und Variablen)