



Informatik für Schüler, Foliensatz 22

Wiederholung

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal
30. April 2009

Grundbegriffe der objektorientierten Programmierung

```
class Klassenname():  
    {Attribut = Wert}  
    {def Methode(self {,arg}):  
        Anweisung  
        {Anweisung}}
```



```
Objektname = Klassenname()  
Objektname.Methode(Argumente_außer_self)  
a = Objektname.Attribut
```

Was ist:

- ein Attribute, eine Methode, eine Klasse
- ein Objekt
- das Argument »self«?

Welche Fehler enthält die Klassendefinition

```
class text():  
    # Attribute  
    L = []  
    s=''  
    def append(self, txt):  
        s += txt  
    def insert(self, txt, pos):  
        self.s = self.s[:pos] + txt + self.s[pos:]  
    def show():  
        print s
```

- Was ist an folgenden Anweisungen falsch?

```
a=text  
a.append(self, 'Katze')
```



Eine Zeichenfeldklasse

```
class ZF2D():
```

Attribute

```
    zf = []  
    z = 0 # Zeilenanzahl  
    s = 0 # Spaltenanzahl
```

Methoden

- Erweiterung der leeren Liste »zf« zu einer 2D-Liste (siehe Foliensatz 13, Folie 6):

```
def create(self, Zeilen, Spalten, Zeichen):  
    self.s = Spalten  
    self.z = Zeilen  
    for idx in range(Zeilen):  
        self.zf.append(Spalten*[Zeichen])
```



- Umwandeln in eine Zeichenkette

```
def to_str(self):  
    s = ''  
    for Zeile in self.zf:  
        for Zeichen in Zeile:  
            s += Zeichen  
        s += '\n'  
    return1 s
```

- Anzeigen des Zeichenfelds

```
def show(self):  
    print '\nZeichenfeld ', self.s, 'x', self.z  
    print self.to_str()2
```

¹auch Methoden können einen Rückgabewert haben

²Aufruf einer eigenen Methode der Klasse

■ Zeichnen eines Punktes

```
def drawPoint(self, x, y, Zeichen):  
    s = int3(round4(x))  
    z = int(round(y))  
    if 0<=s and s<self.s and 0<=z and z<self.z:5  
        self.zf[z][s] = Zeichen
```

³Umwandlung in »int«, Feldindex muss Typ »int« haben

⁴es soll auch float als Eingabe zugelassen sein; »float« muss auf ganze Zahl gerundet werden; konvertiert auch »int« nach »float«

⁵Kontrolle, dass der Punkt im Zeichenfeld liegt

- Zeichnen einer Linie mit den Endpunkten $\mathbf{a} = (a_x, a_y)$ und $\mathbf{b} = (b_x, b_y)$

```
def drawLine(self, ax, ay, bx, by, Zeichen):  
    if abs(ax-bx)>abs(ay-by)6:  
        for x in range(int(round(ax)), int(round(bx))):  
            y = (ay + (by-ay)/(float7(bx)-ax)*(x-ax))  
            self.drawPoint(x, y, Zeichen)  
    else:  
        for y in range(int(round(ay)), int(round(by))8):  
            x = (ax + (bx-ax)/(float(by)-ay)*(y-ay))  
            self.drawPoint(x, y, Zeichen)
```

⁶Anstieg kleiner 1: Wiederhole von der Anfangs bis zur Endspalte:
Berechne y

⁷Stellt sicher, dass der Anstieg als Gleitkommazahl berechnet wird,
ganzzahlige Rechnung ergibt Null

⁸Anstieg größer 1: Wiederhole von der Anfangs- bis zur Endzeile:
Berechne x



Test der Klasse

```

a=ZF2D()
a.create(7, 40, '*')
a.drawPoint(1, 30, 'U')
a.drawLine(3, 2, 34, 8, 'x')
a.show()

```

■ Ausgabe

Zeichenfeld 7 x 40

```

*****
*****U*****
**XXXX*****
*****XXXXXXXX*****
*****XXXXXXXX*****
*****XXXXXXXX*****
*****XXX*****

```




- Zeichenfeld erzeugen
- Wiederhole für alle Linienanfangspunkte (auch die nicht sichtbaren)
 - Zeichne eine Linie mit dem Entpunkt in der untersten Zeile 20 Spalten weiter hinten
 - Zeichne eine Linie mit dem Entpunkt in der untersten Zeile 20 Spalten weiter vorn



22.3: Textfeld

- Erweitern Sie die Klassendefinition um eine Methode

```
def drawText(self, x, y, Text)
```

die ab der Position (x, y) einen Text in des Zeichenfeld schreibt.

- Schreiben Sie eine Testprogramm, das die Werte des Tupels

```
t = ((1, 23), (2, 48), (3, 12345));
```

mit einer Schleife wie folgt tabellarisch darstellt:

```
Zeichenfeld 10 x 40
```

Eingabe	Ausgabe
1	23
2	48
3	12345