Zwischenprüfung Programmieren II

19.10.2016 von 13:40 bis 14:40

Dauer: 45 Minuten
Erlaubte Hilfsmittel:

- Skript Programmierung II
- Weitere Unterlagen (Bücher, Notitzen etc.)

Laptops/Taschenrechner sind nicht erlaubt!

Name: Micha Meier

Punkte: 28.5

Note: 5.5

Aufgabe 1 (4 Punkte)

Was sind docstrings und weshalb sind diese in Python wichtig ?

Schreiben Sie ein konkretes Beispiel einer Klasse, welche docstrings verwendet (mind. 1 Methode)

Sie beschreiben die Funktion / Klasse / Methode. → Was wird gemacht und welche Parameter wie eingegeben werden müssen. Sie sind vorallem dann von Nutzen / wichtig wenn mehrere Leute an demselben Code arbeiten. So weiss jeder was im Code gemacht wird.

Bsp:

```
class Punkt:                                   Klassen-docstring
    def __init__(self, x=0, y=0):
        """
        Dies ist ein Punkt im R^2
        :param x: X-Koordinate (float)
        :param y: Y-Koordinate (float)
        """
        self.x = x
        self.y = y
```

Eingerückt → Docstring

3

Aufgabe 2 (6 Punkte)

Gegeben ist die folgende Klasse „Zahl":

```
class Zahl:
    def __init__(self, z):
        self.wert = z

    def __add__(self, other):
        return N(self. wert + other.wert)

    def __sub__(self, other):
        return N(self. wert - other.wert)

    def __str__(self):
        return str("(" + str(self.wert) + ")")

    def __float__(self):
        return float(self.wert)

    def __int__(self):
        return int(self. wert)
```

a) Was repräsentiert diese Klasse ?

b) Weshalb sind die folgende Codezeilen möglich und was geschieht da genau ? Was sind r0 und r1 (Typ und Wert) ?

```
r0 = Zahl(400) - Zahl(200)
r1 = Zahl(3.3) + Zahl(1.2)
```

c) Was geschieht hier ?

```
s = str(r)
i = int(r)
f = float(r)
```

a.) eine beliebige Zahl → Struktur und noch keine Zahl als Objekt selbst (erst bei einer Erstellung einer Instanz)

b.) Die Addition und Subtraktion der Instanzen sind durch die magischen ~~Funktionen~~ Methoden __add__(self, other): und __sub__(self, other): möglich. Dabei ist das jeweilige Resultat wieder eine beliebige Zahl die aber in der Klasse "N" aufgerufen wird → z.B: class N: """Operationen mit natürlichen Zahlen"""
r0 ergibt 200 → Integer Typ: Zahl
r1 ergibt N(4.5) → je nach Klasse N gibt es einen ValueError weil 4.5 keine natürliche Zahl ist oder es wird auf eine natürliche Zahl umgewandelt → mit int(r1)
Dann: r1 ergibt 5, Typ = Integer

c.) • s = str (r) # r = Zahl (100) → Umwandlung zu String
→ Ausgabewert: "(200)" (magische Methode __str__ wird aufgerufen)

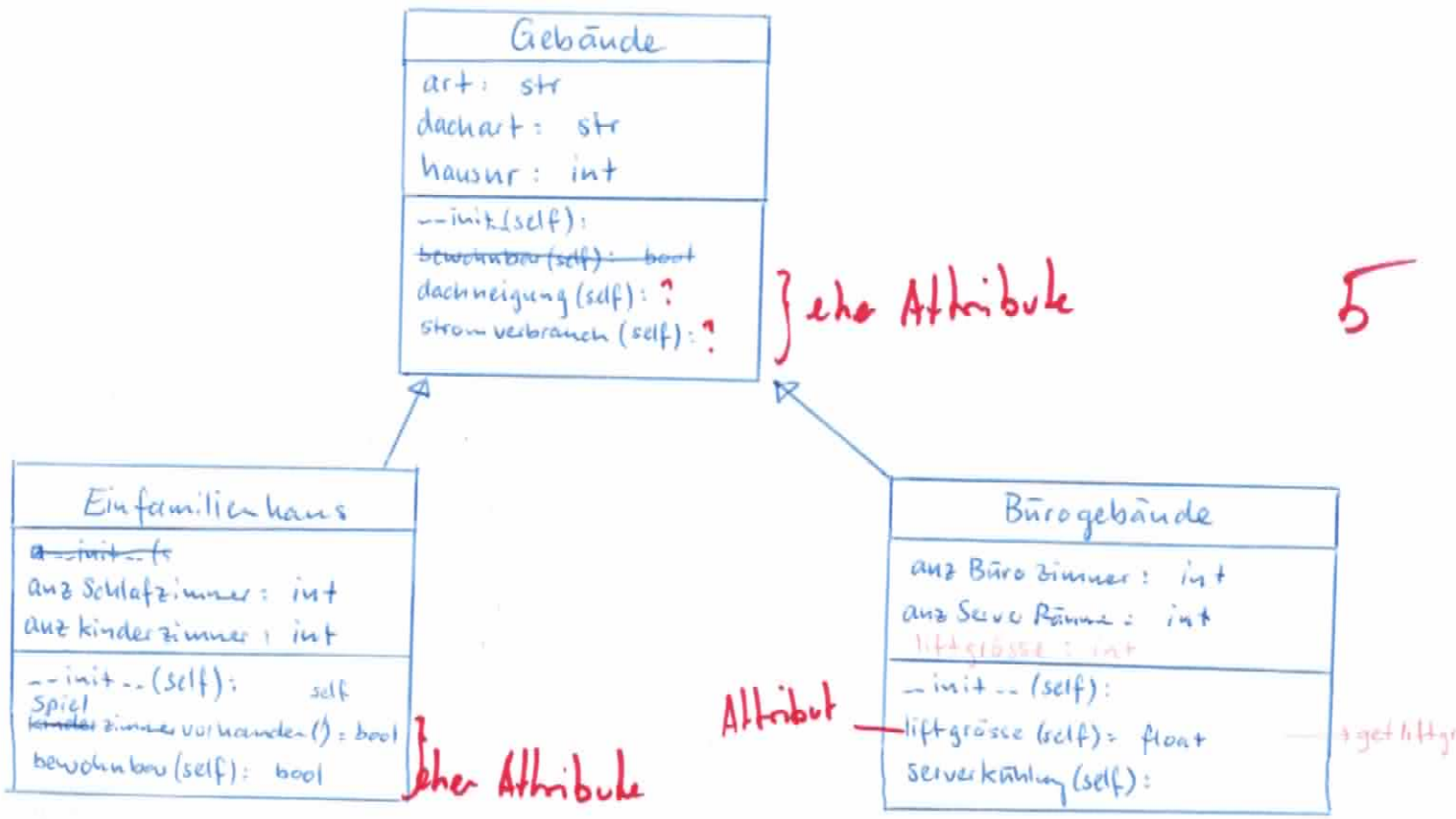
• i = int (r) # r = Zahl (200) → Umwandlung zu Integer
Ausgabewert: 200 (magische Methode __int__ aufgerufen)
r = Zahl (4.5)
Ausgabewert: 4

6

• f = float (r) # r = Zahl (200) → Umwandlung zu Floating Point
Ausgabewert: 200.0 (magische Methode __float__ aufgerufen)
r = Zahl (4.5)
Ausgabewert: 4.5

Aufgabe 3 (6 Punkte)

Die Klassen „Einfamilienhaus" und „Bürogebäude" werden von der Klasse „Gebäude" vererbt. **Zeichnen Sie ein UML Klassendiagramm** und erstellen Sie sinnvolle Attribute (min. 2 pro Klasse) und Methoden (min. 3 pro Klasse mit Konstruktor).



Aufgabe 4 (6 Punkte)

Erstellen Sie eine Klasse Vector3, welche einen dreidimensionalen Vektor repräsentiert.

Über den Konstruktor werden die Komponenten x, y und z definiert. Wird nichts angegeben, so wird ein Null-Vektor erstellt.

Die **Addition** und **Subtraktion** von Vektoren soll dabei ermöglicht werden.

Es soll auch möglich sein den Vektor über die `print(...)` Funktion auszugeben.

Beispiel:

```
v0 = Vector3(3,4,1)
v1 = Vector3()
v2 = v0 + v1

v3 = Vector3(3,4,2) - Vector3(3,1,2) + Vector3(2,1,9)

print(v3)
```

Docstring:

```
"""
Vektoroperationen
:param x: x-Komponente
:param y: y-Komponente
:param z: z-Komponente
"""
```

```
class Vektor3:
    def __init__(self, x=0, y=0, z=0):
        self.x = x
        self.y = y
        self.z = z
    def __add__(self, other):
        return Vektor3(self.x + other.x, self.y + other.y, self.z + other.z)

    def __sub__(self, other):
        return Vektor3(self.x - other.x, self.y - other.y, self.z - other.z)

    def __str__(self):
        return '(' + str(self.x) + ',' + str(self.y)
               + ',' + str(self.z) + ')'
```

5

Aufgabe 5 (4 Punkte)

a) Was ist eine Klassendefinition ?

b) Was ist eine Instanz einer Klasse ?

a.) Sie beschreibt die Struktur eines ~~Objektes~~ Objekttyps, welches aus Attributen und Methoden besteht
→ die Klasse ist noch kein Objekt
Bauplan der Klasse

3.5

b.) Das eigentliche Objekt. Aus einer Klasse können zahlreiche Objekte / Instanzen erstellt werden

Aufgabe 6 (6 Punkte)

Erstellen Sie mit PyQT4 eine GUI, welche folgendermassen aussieht:


```
import sys
from PyQt4.QtCore import *
from PyQt4.QtGui import *
```

```
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Adresseingabe")

        layout_v = QVBoxLayout()
        ~~layout_h = QHBoxLayout()~~      # nicht notwendig

        name = QLabel("Name:")
        nameline = QLineEdit()
        adresse = QLabel("Adresse:")
        adressetext = QTextEdit()
        button = QPushButton("Ok")

        layout_v.addWidget(name)
        layout_v.addWidget(nameline)
        layout_v.addWidget(adresse)
        layout_v.addWidget(adressetext)
        layout_v.addWidget(button)

        center = QWidget()
        center.setLayout(layout_v)

        self.setCentralWidget(center)
        self.show()

def main():
    app = QApplication(sys.argv)
    main window = MyWindow()
    main window.raise()
    app.exec_()
if __name__ == '__main__':
    main()
```

6