Aufgabe 1 (4 Punkte)

a) Was ist eine Klassendefinition ?

b) Was sind Instanzen ?

Beschreiben Sie das Konzept und schreiben Sie eine Beispielklasse bei der dieses gezeigt wird.

4

Klassendefinition:
Eine Klassendefinition oder Klasse beschreibt die Struktur eines Objektes welche aus Attributen und Methoden besteht. Es ist nur die Beschreibung eines Objekttyps, jedoch nicht das Objekt

Instanzen: Mit der Instanzen ruft man die Methode der Klasse auf. Es ist die Realisierung einer Klassendefinition mit eigenen ~~klassen~~ Werten der Klasserattribute

Klassendefinition

```
class Fenstes:
    def __init__(self, laenge, breite):
        self.laenge = laenge
        self.breite = breite

    def __str__(self):
        return f"{self.laenge}, {self.breite}"
```

Instanzen

```
a = Fenstes(2.5, 3)
print(a)
```

Aufgabe 2 (4 Punkte)

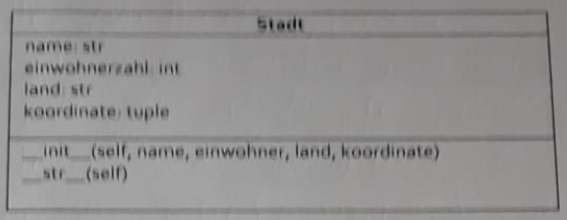
Was sind magische Methoden in Python und für was können diese verwendet werden ?

Sind spezielle Methoden, welche einer Klasse besondere Fähigkeiten geben. Sie werden mit zwei Unterstrichen vor und hinter der Methode dargestellt.
z.B. __str__ => Eine Klasse in eine Zeichenkette umgewandelt. Sie sind implizit, also werden sie automatisch abgerufen bei bestimmten definitionen in den Instanzen. Dies ermöglicht einfacheres program

4

Aufgabe 3 (4 Punkte)

a) Implementieren Sie eine Klasse `Stadt` in Python. Diese ist durch folgendes UML Diagramm gegeben:



4

b) «koordinate» und «land» sind vom Datentyp „tuple" resp. „str". Weshalb wäre es sinnvoll für diese eigene Klassen einzuführen ?

a)
```
class Stadt:
    def __init__(self, name, einwohner, land, koordinate):
        self.name = name
        self.einwohnerzahl = einwohner
        self.land = land
        self.koordinate = koordinate

    def __str__(self):
        return f"Stadt: {self.name}, Einwohnerzahl: {self.einwohner}, Land: {self.land}, Koordinate: {self.koordinate}"
```

b) - Wenn man Koordinaten in einer seperaten Klasse definiert könnte man einfacher die Bedingungen festlegen, sowie die Form. Bsp. Schweizer Landeskoord müssen 7 Stellen haben und sind ein tuple.

- Bei Land ist das Problem, dass man wenn man 1 zu n Beziehungen hat. Bei der oberen Darstellu jedes einzelne Attribut das, dass gleiche Land beschrie ändern muss. Mit Klasse muss man es nur einma ändern.

Aufgabe 4 (6 Punkte)

Zutreffendes Ankreuzen. (Falschantworten -1P, richtige Antworten 1P). Minmalpunktzahl 0.

4.1 Was wird automatisch aufgerufen, wenn eine neue Instanz einer Klasse erstellt wird ?

- ☒ Der Konstruktor
- ☐ Die Klasse
- ☐ Eine None-Methode
- ☐ Eine Methode, welche einen Wert zurückgibt

1

4.2 Was ist die Ausgabe dieses Progamms?

```
class Fenster:
    def __init__(self, id):
        self.id = id
        id = 100

k = Fenster(200)
print(k.id)
```

- ☐ Das Programm stürzt ab
- ☐ 100
- ☒ 200
- ☐ anderer Wert

1

4.3 Welche Aussagen sind wahr?

- ☐ Klassen gibt es nicht in Python
- ☒ `int(3).__add__(2)` gibt 5
- ☒ Es können mehrere Instanzen aus derselben Klasse erzeugt werden
- ☒ Wird eine Klasse vererbt, so muss der Konstruktor der vererbten Klasse mit `super().__init__()` manuell aufgerufen werden, damit dessen Attribute korrekt initialisiert werden.

3

4.3 Was ist die Ausgabe dieses Programms?

```
class Test:
    def __init__(self, value=10):
        self.q = value
    def __str__(self):
        return f"{self.q}"
print(Test(5))
```

- ☐ Das Programm stürzt ab
- ☐ 10
- ☒ 5
- ☐ None

1

Aufgabe 5 (6 Punkte)

Gegeben ist die Funktion `int_to_roman(num)` welche einen Integer in eine Römische Zahl (Zeichenkette) umwandelt. Gegeben ist eine weitere Funktion `roman_to_int(s)`, welche eine Römische Zahl (Zeichenkette) in einen Integer umwandelt:

Floss!

```
def int_to_roman(num):
    val = [
        1000, 900, 500, 400,
        100, 90, 50, 40,
        10, 9, 5, 4,
        1
        ]
    syb = [
        "M", "CM", "D", "CD",
        "C", "XC", "L", "XL",
        "X", "IX", "V", "IV",
        "I"
        ]
    roman_num = ''
    i = 0
    while  num > 0:
        for x in range(num // val[i]):
            roman_num += syb[i]
            num -= val[i]
        i += 1
    return roman_num
```

```
def roman_to_int(s):
    rom_val = {'I': 1, 'V': 5, 'X': 10, 'L': 50,
'C': 100, 'D': 500, 'M': 1000}
    int_val = 0
    for i in range(len(s)):
        if i > 0 and rom_val[s[i]] >
                        rom_val[s[i - 1]]:
            int_val += rom_val[s[i]] - 2 *
                        rom_val[s[i - 1]]
        else:
            int_val += rom_val[s[i]]
    return int_val
```

Schreiben Sie eine Klasse «Roman», welche – unter Verwendung von Magischen Methoden Römische Zahlen addieren kann. Mit der Magischen Methode `__str__` kann diese auch in eine Zeichenkette umgewandelt werden. Ausgabe ist wiederum römisch. Mit der Umwandlung zu Integer (magische Methode `__int__(...)`) soll die Zahl als Integer konvertiert werden können:

Mit der Klasse soll also folgendes möglich sein:

```
print(Roman("MM") + Roman("X"))
b = Roman("MMXIX")
print(int(b))
print(int(Roman("X") + Roman("I")))
```

Verwenden Sie dazu die oben definierten Funktionen direkt in der Klasse.
(nicht abschreiben, direkt aufrufen!!)

```
class Roman:
    def __init__(self, roman):
        self.roman = roman

    def __add__(self, other):
        return self.int_to_Roman(self.roman_to_int(self.roman) + self.roman_to_int(other.roman))

    def __int__(self):
        return self.roman_to_int(self.roman)

    def __str__(self):
        return f"{self.roman}"

    def int_to_Roman(num):
        ...

    def roman_to_int(s):
        ...
```

Aufgabe 6 (6 Punkte)

Erstellen Sie eine Klasse «Student». Diese soll mit Name, Vorname, Geschlecht, Immatrikulationsnummer (alles Zeichenketten) initialisiert werden können.

Implementieren Sie folgende Methoden:

`setAge(...)` : Setzt das Alter
`setMark(...)` : Setzt Note in einem Fach.
Es sollen beliebig viele Fächer möglich sein.
(Tipp: Dictionary)
`display(...)` : gibt alle Informationen aus

6

```
class Student:
    def __init__(self, name, vorname, gender, Imm_nr, age=0):
        self.name = name
        self.vorname = vorname
        self.gender = gender
        self.Imm_nr = Imm_nr
        self.setAge(age)
        self.mark = {}

    def setAge (self, age):
        self.age = age

    def setMark(self, topic, mark):
        self.mark["topic"] = mark

    def display (self):
        print (self.name, self.vorname, s
        self.Imm_nr, self.age, self.mark)
```