Python (OOP)

Prüfungsvorbereitung

Aufgabe 1

Erkläre die folgenden Begriffe der objektorientierten Programmierung.

(a) Klasse

(b) Instanz (oder Objekt)

(c) Objekteigenschaft

(d) Objektmethode

(e) Klasseneigenschaft

(f) Klassenmethode

(g) Konstruktor

Aufgabe 1

(a) Ein Bauplan für Objekte (ein abstrakter Datentyp)

(b) Ein Objekt, das zur Laufzeit aus einer Klasse erzeugt wird

(c) Eine Variable, die zu einem Objekt gehört

(d) Eine Funktion, die zu einem Objekt gehört

(e) Eine Variable, die zu einer Klasse gehört

(f) Eine Funktion, die zu einer Klasse gehört

(g) Eine spezielle Methode, mit der ein Objekt initialisiert wird.

Aufgabe 2

```
1  class MyClass:
2
3      def __init__(self, a, b):
4          self.a = a
5          self.b = b
6
7      def d(self, c):
8          return (self.a + self.b + c)
9
10 x = MyClass(3, 4)
11 print(x.d(5))
```

Aufgabe 2

```
12
```

Aufgabe 3

```
1  class Aufgabe():
2
3      def __init__(self, a=3, b=2):
4          self.x = b
5          self.y = a
6
7      def __str__(self):
8          return '({0.y}, {0.x})'.format(self)
9
10 print(Aufgabe(4))
```

Aufgabe 3

```
(4, 2)
```

Aufgabe 4

```
1  class Vektor():
2
3      def __init__(self, x=0, y=0):
4          self.x = x
5          self.y = y
6
7      def __str__(self):
8          return '({0.x},{0.y})'.format(self)
9
10     def __add__(u, v):
11         return  Vektor(u.x+v.x, u.y+v.y)
12
13     def __sub__(u, v):
14         return Vektor(u.x-v.x, u.y-v.y)
```

```
15
16      def __rmul__(u, k):
17          return Vektor(k*u.x, k*u.y)
18
19      def betrag(self):
20          return (self.x**2 + self.y**2)**0.5
21
22  a = Vektor(2,3)
23  b = Vektor(4,7)
24  c = 2*a-b
25  print(c.betrag())
```

Aufgabe 4

```
1.0
```

Aufgabe 5

Welche Ausgabe macht das folgende Programm?

```
1  class Parent():
2      def __init__(self, a):
3          self.a = a
4      def m(self, x):
5          return (x*self.a)
6
7  class Child(Parent):
8      def __init__(self, a, b):
9          super().__init__(a)
10         self.b = b
11     def m(self, y):
12         return (self.a + self.b + super().m(y))
13
14 c = Child(2,3)
15 print(c.m(10))
```

Aufgabe 5

25

Aufgabe 6

Implementiere die Klasse `Lampe`, welche das Verhalten einer Lampe aufgrund des folgenden Klassendiagramms modelliert.

`Lampe`
`zustand: bool`

`Lampe()`
`schalten()`
`str(): str`

Hinweise:

- Der Konstruktor erzeugt eine ausgeschaltete Lampe.
- Die Instanzvariable `zustand` kann nur die Werte `True` oder `False` annehmen.
- Die Spezialmethode `str()` wird mit Hilfe von `__str__` implementiert und hat je nach Zustand der Lampe den Rückgabewert `'Lampe ein'` bzw. `'Lampe aus'`.

Aufgabe 6

```
1  class Lampe:
2
3      def __init__(self):
4          self.zustand = False
5
6      def __str__(self):
7          if self.zustand == True:
8              return 'Lampe an'
9          else:
10             return 'Lampe aus'
11
12     def schalten(self):
13         if self.zustand == False:
14             self.zustand = True
15         else:
16             self.zustand = False
```

```
18 # Test-Client: (nur zur Illustration)
19 L = Lampe()  # erzeugt einen neue (ausgeschaltete) Lampe
20 L.schalten() # Schaltet die Lampe ein
21 L.schalten() # Schaltet die Lampe aus
22 L.schalten() # Schaltet die Lampe ein
23 print(L)     # => 'Lampe ein'
```

Aufgabe 7

Zähle zwei Vorteile der objektorientierten Programmierung auf.

Aufgabe 7

Hier sind vier Vorteile:

(a) Die Objektorientierung ist an die Sichtweise der Menschen angelehnt. Sie schafft eine Abstraktion, durch die wir Programme besser verstehen und einfacher programmieren können.

(b) Durch Vererbung kann Code in anderen Klassen wiederverwendet werden, was effizient ist.

(c) Polymorphie („Vielgestaltigkeit“): Variablen- und Funktionen können in unterschiedlichen Klassen den gleichen Namen haben, ohne das Konflikte enstehen.

(d) OOP unterstützt *Information Hiding*. Das bedeutet, dass Änderungen an den Objekten nur über dafür vorgesehene Methoden (sog. Schnittstellen) möglich sind. Objekte