1. (3 Punkte) Die Klasse Ding hat die Attribute name und wert. Implementiere die Klasse Ding. Folgender Dialog soll möglich sein:

```
d = Ding("A",100)
print(d)
```

```
Das Ding A hat den Wert 100
```

Lösung:

```
class Ding:
    def __init__(self,name,wert):
        self.name = name
        self.wert = wert

    def __str__(self):
        return "Das Ding {:} hat den Wert {:}".format(self.name,self.wert)
```

2. (3 Punkte) Die Klasse Zutat hat die Attribute bezeichnung und menge. Implementiere die Klasse Zutat. Folgender Dialog soll möglich sein:

```
z = Zutat(800,"Mehl")
print(z)
```

```
Es werden 800g Mehl benötigt.
```

Lösung:

```
class Zutat:
    def __init__(self,menge,bezeichnung):
        self.menge = menge
        self.bezeichnung = bezeichnung

    def __str__(self):
        return "Es werden {:}g {:} benötigt.".format(self.menge,self.bezeichnung)
```

3. (3 Punkte) Schreibe Code, der den folgenden Dialog möglich macht:

```
a = A('H')
b = A('C')
print(a.aba, a.d)
print(b.aba, b.d)
```

```
17 H
17 C
```

Lösung:

```
class A:
    def __init__(self, d):
        self.d = d
        self.aba = 17
```

4. (3 Punkte) Schreibe Code, der den folgenden Dialog möglich macht:

```
k1 = K(1)
k2 = K(2)
print(k1.x, k1.y, k2.x, k2.y)
```

```
1 1 2 2
```

Lösung:

```
class K:
    def __init__(self, x):
        self.x = x
        self.y = x
```

5. (4 Punkte) Die Klasse A hat ein Attribut zahl. Dem Konstruktor von A wird der Wert für das Attribut zahl übergeben. Die Klasse B erbt von A und erhält ein weiteres Attribut zahl2. Dem Konstruktor von B werden Werte für beide Attribute übergeben, der Konstruktor von B nutzt den Konstruktor von A. Implementiere die Klasse B.

Lösung:

```
class B(A):
    def __init__(self,zahl,zahl2):
        super().__init__(zahl)
        self.zahl2 = zahl2
```

6. (3 Punkte) Die Klasse A hat eine Methode berechne, die einen int-Wert zurückgibt. Die Klasse B erbt von A und überschreibt die Methode berechne. Sie addiert noch 10 auf das Ergebnis von A drauf. Implementiere die Klasse B.

Lösung:

```
class B(A):
    def berechne(self):
        return super().berechne() + 10
```

7. (3 Punkte) Die Klasse Konto hat das Attribut kontostand. Beim Erstellen eines Kontos wird der Wert von Kontostand auf 0 gesetzt. Implementiere die Klasse Konto unter Wahrung des Geheimnisprinzips mit getter und setter-Methode.

Lösung:

```
class Konto:
    def __init__(self):
        self.__kontostand = 0

    def setKontostand(self,x):
        self.__kontostand = x

    def getKontostand(self):
        return self.__kontostand
```

8. (3 Punkte) Die Klasse Person hat das Attribut alter. Dem Konstruktor von Person wird der Wert von alter übergeben. Implementiere die Klasse Person unter Wahrung des Geheimnisprinzips mit getter und setter-Methode, wobei allen Unterklassen von Person der direkte Zugriff auf das Attribut alter möglich sein soll.

Lösung:

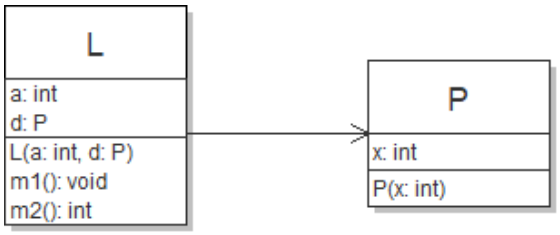
```
class Person:
    def __init__(self, alter):
        self._alter = alter

    def setAlter(self,x):
        self._alter = x

    def getAlter(self):
        return self._alter
```

9. (6 Punkte) a. Gib für die beiden Klassen ein passendes Klassendiagramm an.

   b. Was erscheint auf der Konsole?

```
class L:
    def __init__(self,a,d):
        self.a = a
        self.d = d

    def m1(self):
        self.a = self.d.x + 1

    def m2(self):
        return self.a

class P:
    def __init__(self,x):
        self.x = x

p = P(1)
k = L(2,p)
print(k.m1())
print(k.m2())
```

Lösung:



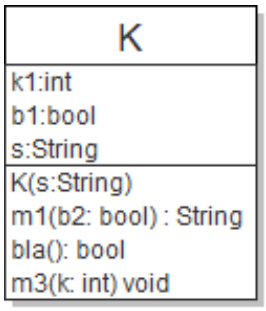
b. None, 2

10. (6 Punkte) a. Gib für die Klasse ein passendes Klassendiagramm an.

    b. Was erscheint auf der Konsole?

```
class K:
    def __init__(self,s):
        self.k1 = 1
        self.b1 = False
        self.s = s

    def m1(self, b2):
        if b2:
            return self.s + '#'
        else:
            return self.s + '?'

    def bla(self):
        return not self.b1

    def m3(self, k):
        self.k1 += k
        self.b1 = not self.b1

    def __str__(self):
        return str(self.k1) + ' ' + str(self.b1) + ' ' + self.s

k = K('hi')
k.m3(4)
temp = k.m1(k.bla())
k.s = temp
print(k)
```

Lösung:



b. 5 True hi?

11. (8 Punkte) a. Implementiere eine Klasse `Buch` mit den Attributen `autor` und `titel`.
b. Erzeuge zwei Buch-Objekte `b1` und `b2`: *Homo faber* von Max Frisch und *Buddenbrooks* von Thomas Mann.
c. Erstelle eine Liste `buecher` die `b1` und `b2` enthält.
d. Erstelle ein Dictionary `buchmap` mit dem Titel eines Buches als key und dem Buch-Objekt als value. `buchmap` soll zu Beginn Einträge für `b1` und `b2` enthalten.
e. Erweitere `buchmap` um einen Eintrag: *Agnes* von Peter Stamm.
f. In `buchmap` seinen noch mehrere Bücher eingefügt worden. Welcher Ausdruck mit `buchmap` liefert den Autor von `Der Prozess` (wir nehmen an, dass das entsprechende Buch in `buchmap` vorhanden ist).

Lösung:

```
class Buch:
    def __init__(self,autor,titel):
        self.autor = autor
        self.titel = titel

b1 = Buch("Max Frisch","Homo faber")
b2 = Buch("Thomas Mann","Buddenbrooks")
buecher = [b1, b2]
buchmap = {"Homo faber":b1, "Buddenbrooks":b2}
buchmap["Agnes"] = Buch("Peter Stamm","Agnes")
buchmap["Der Prozess"].autor
```

12. (9 Punkte) a. Implementiere eine Klasse `Filmmusik` mit den Attributen `film` und `komponist`, so dass die folgenden Filmmusik-Objekte erzeugt werden können:

```
m1 = Filmmusik("Jurassic Park","John Williams")
m2 = Filmmusik("Star Wars","John Williams")
m3 = Filmmusik("Terminal","John Williams")
m4 = Filmmusik("Inception","Hans Zimmer")
m5 = Filmmusik("Interstellar","Hans Zimmer")
m6 = Filmmusik("Lord of the Rings","Howard Shore")
```

b. Implementiere eine string-Methode, die mit `print(m1)` folgende Ausgabe erzeugt:
`Jurassic Park - Musik: John Williams`
c. Erzeuge eine Liste `fmlist` mit den Objekten m1, m2... m6. Die Objekte m1 bis m6 können als erzeugt vorausgesetzt werden.
d. Gehe mit einer Schleife durch die `fmlist` und gib alle Filmtitel mit Musik von John Williams aus.
e. Erzeuge für die 6 Filme in `fmlist` ein Dictionary `fmmap`, bei dem jedem Komponistennamen (key) eine Liste seiner Filmmusik-Objekte als value zugeordnet ist.

Beispiel einer Nutzung von `fmmap`:

```
>>> for f in fmmap["Hans Zimmer"]:
    print(f)

Inception - Musik: Hans Zimmer
Interstellar - Musik: Hans Zimmer
```

Lösung:

```
class Filmmusik:
    def __init__(self,film,komponist):
        self.film = film
        self.komponist = komponist

    def __str__(self):
        return self.film + " - Musik: " + self.komponist


fmlist = [m1,m2,m3,m4,m5,m6]
for f in fmlist:
    if f.komponist == "John Williams":
        print(f.film)

fmmap =  {"John Williams":[m1,m2,m3],
          "Hans Zimmer":[m4,m5],"Howard Shore":[m6]}
```

13. (8 Punkte) Welche Ausgabe erscheint jeweils auf der Konsole?

a.
```
class Clock:
    def __init__(self, time):
        self.time = time
    def print_time(self):
        time = '6:30'
        print(self.time)

clock = Clock('5:30')
clock.print_time()
```

b.
```
class Clock:
    def __init__(self, time):
        self.time = time
    def print_time(self, time):
        print(time)

clock = Clock('5:30')
clock.print_time('10:30')
```

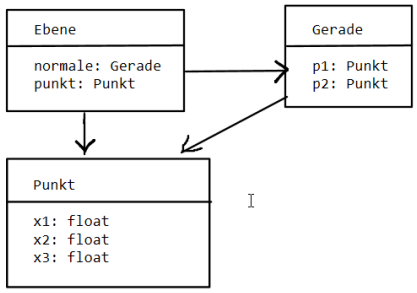
c.
```
class Clock:
    def __init__(self, time):
        self.time = time
    def print_time(self):
        print(self.time)

boston_clock = Clock('5:30')
paris_clock = boston_clock
paris_clock.time = '10:30'
boston_clock.print_time()
```

Lösung:

a. 5:30
b. 10:30
c. 10:30

14. (3 Punkte) Es sollen Objekte aus dem Bereich der Vektorgeometrie modelliert werden. Eine Ebene, eine Gerade und ein Punkt. Ein Punkt soll durch drei float-Werte `x1, x2, x3` festgelegt sein. Eine Gerade soll durch zwei Punkte bestimmt sein, eine Ebene soll durch eine Gerade (ihre Normale) und einen Punkt definiert sein. Zeichne ein passendes Klassendiagramm mit den Klassennamen und den Attributen.

Lösung:



15. (3 Punkte) Es sollen Objekte aus dem Bereich der Vektorgeometrie modelliert werden. Eine Ebene, eine Gerade und ein Punkt. Ein Punkt soll durch drei float-Werte `x1, x2, x3` festgelegt sein. Eine Gerade soll durch zwei Punkte `p1` und `p2` bestimmt sein, eine Ebene soll durch 3 Punkte `a, b, c` definiert sein. Zeichne ein passendes Klassendiagramm mit den Klassennamen und den Attributen.

Lösung: