

Dozent: Michael Pfeuti

Übung 5: Klassen & Vererbung

I Biblothek

Wir wollen eine einfach Bibliothek mit Büchern programmieren.

1 Klasse: Buch

Wir brauchen eine Klasse Buch mit folgenden Eigenschaften/Attributen

- Titel des Buches
- Autor des Buches
- Einband (Hardcover oder Paperback)
- · Genre des Buches
- Seiten mit Text (eine Liste von Strings)

Diese Eigenschaften sollen dem Konstruktor als Argumente übergeben werden. Die Klasse Buch hat drei Interaktionen/Methoden:

- öffnen und schliessen der Zustand, ob das Buch offen oder geschlossen ist, soll in einem Attribut gespeichert werden. Am Anfang soll das Buch geschlossen sein, d.h. im Konstruktor soll dieses Attribut entsprechende gesetzt werden. Wenn ein Buch offen (respektive geschlossen) ist, und es wird nochmals geöffnet (respektive geschlossen) soll eine passende Nachricht auf der Kommandozeile ausgegeben werden.
- lesen bei jedem Aufruf von dieser Funktion soll der Inhalt der nächsten Seite auf der Kommandozeile ausgegeben werden. Dazu muss in einem Attribut gespeichert werden, welche Seite gelesen wird. Ein Buch muss zuerst geöffnet werden bevor es gelesen werden kann. Beim Lesen soll dies überprüft werden, und falls das Buch noch nicht geöffnet wurde, soll eine Fehlermeldung auf der Kommandozeile ausgegeben werden. Wenn in einem Buch gelesen wurde und es wird geschlossen, so soll nach dem nächsten Öffnen, die Methode *lesen* wieder am Anfang beginnen. Wenn die letzte Seite gelesen wurde, soll das als Information auf der Kommandozeile ausgegeben werden, und das Buch automatisch geschlossen werden.

Weiter soll die Magic Method __str__ auf der Buch Klasse implementiert werden. Diese Methode soll einen String zurückgeben, welcher den Zustand (aber nicht den Text/Inhalt) des Buches als String darstellt. Der Rückgabewert der Funktion soll folgende Information enthalten: Buchtitel, Autor, Einband, Genre, ist Buch offen (wenn ja bis, wo wurde gelesen)?.



Dozent: Michael Pfeuti

Übung 5: Klassen & Vererbung

```
__init__(self, titel, autor, einband, genre, seiten):
self.titel = titel
self.autor = autor
                 self.einband = einband
                 self.genre = genre
                 self.geoeffnet = False
self.aktuelle_seite = 0
def __str__(self):
    zustand = "geöffnet" if self.geoeffnet else "geschlossen"
                lesefortschritt = f", gelesen bis Seite (self.aktuelle_seite)" if self.geoeffnet and self.aktuelle_seite > 0 else ""
return f"Buchtitel: {self.titel}, Autor: {self.autor}, Einband: {self.einband}, Genre: {self.geore}, Zustand: {zustand}{lesefortschritt}.
                 if self.geoeffnet:
                     print("Das Buch ist bereits geöffnet.")
                     self.geoeffnet = True
self.aktuelle_seite = 0
                      print("Das Buch ist bereits geschlossen.")
                       self.geoeffnet = False
                      print(f"Das Buch '{self.titel}' wurde geschlossen.")
                 if not self.geoeffnet:
                      if self.aktuelle seite < len(self.seiten):
                          print(self.seiten[self.aktuelle_seite])
                           if self.aktuelle_seite == len(self.seiten):
    print("Das war die letzte Seite. Das Buch wird automatisch geschlossen.")
                                 self.schliessen()
                            print("Das Buch wurde bereits bis zum Ende gelesen und wird geschlossen.")
```

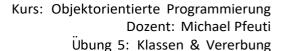
2 Klasse: Bibliothek

Die Klasse Bibliothek verwaltet eine Liste von Büchern, welche ausgeliehen und zurückgegeben werden können. Somit hat diese Klasse zwei Listen. Eine Liste mit dem Buchbestand und eine Liste mit den aktuell ausgeliehenen Büchern.

Die Bibliothek hat drei Interaktionen/Methoden:

- Ein Buch in den Bestand aufnehmen
- Ein Buch ausleihen: Hier soll geprüft werden, dass das Buch nicht bereits ausgeliehen ist, sonst soll eine Fehlermeldung ausgegeben werden.
- Ein ausgeliehenes Buch zurückgeben: Hier soll geprüft werden, dass das Buch zuvor auch ausgeliehen wurde sonst soll eine Fehlermeldung ausgegeben werden.

Weiter soll die Magic Method __str__ auf der Bibliothek Klasse implementiert werden. Es sollen die Titel der Bücher im Bestand im zurückgegeben String stehen. Zudem sollen separat die Titel der ausgeliehenen Bücher im String stehen.





```
class Bibliothek:
         def __init__(self):
             self.buchbestand = []
             self.ausgeliehene_buecher = []
         def buch_aufnehmen(self, buch):
             if buch not in self.buchbestand and buch not in self.ausgeliehene_buecher:
                 self.buchbestand.append(buch)
                 print(f"Buch '{buch.titel}' wurde dem Bestand hinzugefügt.")
                 print(f"Buch '{buch.titel}' befindet sich bereits im Bestand.")
         def buch_ausleihen(self, buch):
             if buch in self.ausgeliehene_buecher:
                 print(f"Buch '{buch.titel}' ist bereits ausgeliehen.")
             elif buch in self.buchbestand:
                 self.buchbestand.remove(buch)
                 self.ausgeliehene_buecher.append(buch)
                 print(f"Buch '{buch.titel}' wurde ausgeliehen.")
                 print(f"Buch '{buch.titel}' befindet sich nicht im Bestand.")
         def buch_zurueckgeben(self, buch):
             if buch in self.ausgeliehene buecher:
                 self.ausgeliehene buecher.remove(buch)
26
                 self.buchbestand.append(buch)
                 print(f"Buch '{buch.titel}' wurde zurückgegeben und dem Bestand hinzugefügt.")
                 print(f"Buch '{buch.titel}' wurde nicht ausgeliehen.")
         def __str__(self):
             bestands_titel = ", ".join(buch.titel for buch in self.buchbestand)
             ausgeliehene_titel = ", ".join(buch.titel for buch in self.ausgeliehene_buecher)
             return f"Bücher im Bestand: {bestands_titel}\nAusgeliehene Bücher: {ausgeliehene_titel}"
```

3 Testlauf

Es soll eine Python-Datei geschrieben werden, welche ausgeführt werden kann und demonstriert, dass alles korrekt programmiert wurde.

- Es sollen 3-5 Bücher erstellt werden und der Bibliothek hinzugefügt werden (d.h. in den Bestand aufnehmen).
- Es sollen Bücher ausgeliehen und zurückgegeben werden, dabei soll auch gezeigt werden, dass kein bereits ausgeliehenes Buch nochmals ausgeliehen werden kann.
- Anhand eines Buches soll gezeigt werden, dass das Lesen des Buches funktioniert.





```
2
3 Excest musen wir die Klassen Buch und Bibliothek definieren, wie oben beschrieben.

2 States Buch
3 Class Buch
5 Class Buch
5 Comment of the Comment of
```

```
print("Sie haben bereits das Ende des Buches erreicht.")
                        self.schliessen()
class Bibliothek:
    def __init__(self):
    self.buchbestand = []
           self.ausgeliehene_buecher = []
     def buch_aufnehmen(self, buch):
           self.buchbestand.append(buch)
print(f"Buch '{buch.titel}' wurde dem Bestand hinzugefügt.")
     def buch_ausleihen(self, buch):
          if buch in self.ausgeliehene_buecher:
           elif buch in self.buchbestand:
                self.buchbestand.remove(buch)
                 self.ausgeliehene_buecher.append(buch)
print(f"Buch '{buch.titel}' wurde ausgeliehen.")
                  print(f"Buch '{buch.titel}' ist nicht im Bestand.")
     def buch zurueckgeben(self, buch):
           if buch in self.ausgeliehene_buecher:
                self.ausgeliehene_buecher.remove(buch)
                  {\tt self.buchbestand.append(buch)}
                 print(f"Buch '{buch.titel}' wurde zurückgegeben.")
                print(f"Buch '{buch.titel}' war nicht ausgeliehen.")
     def __str__(self):
           bestands_titel = ", ".join(buch.titel for buch in self.buchbestand)
ausgeliehene_titel = ", ".join(buch.titel for buch in self.ausgeliehene_buccher)
return f"Bücher im Bestand: {bestands_titel}\nAusgeliehene Bücher: {ausgeliehene_titel}"
if <u>name</u> == "<u>main</u>":
| # Erstellen einiger Bücher
     buch1 = Buch("Python Programmierung", "Max Mustermann", "Hardcover", "Informatik", ["Seite 1 Inhalt", "Seite 2 Inhalt"])
buch2 = Buch("Das Leben des Pi", "Yann Martel", "Paperback", "Abenteuer", ["Seite 1 Inhalt", "Seite 2 Inhalt"])
buch3 = Buch("Der Große Gatsby", "F. Scott Fitzgerald", "Hardcover", "Klassiker", ["Seite 1 Inhalt", "Seite 2 Inhalt"])
```



Dozent: Michael Pfeuti Übung 5: Klassen & Vererbung

```
bibliothek = Bibliothek()
          bibliothek.buch aufnehmen(buch1)
          bibliothek.buch_aufnehmen(buch2)
          bibliothek.buch_aufnehmen(buch3)
          bibliothek.buch ausleihen(buch1)
          bibliothek.buch_ausleihen(buch1) # Versuch, ein bereits ausgeliehenes Buch auszuleihen
          bibliothek.buch_zurueckgeben(buch1)
          bibliothek.buch_zurueckgeben(buch1) # Versuch, ein bereits zurückgegebenes Buch zurückzugeben
          # Demonstration des Lesens eines Buches
          buch1.oeffnen()
          buch1.lesen()
          buch1.lesen()
          buch1.schliessen()
          buch1.lesen() # Versuch zu lesen, nachdem das Buch geschlossen wurde
          print(bibliothek)
109
```



Dozent: Michael Pfeuti

Übung 5: Klassen & Vererbung

II Drucker

In dieser Ü bung wollen wir einen Drucker simulieren. Da es sich um eine Simulation handelt, wird es hier aber keine echten Papier-Ausdrucke geben. Wir werden folgende Drucker Klassen schreiben:

- Schwarz/Weiss Kommandozeilen Drucker
- Farb Kommandozeilen Drucker
- Einzelbuchstaben-Bild Drucker

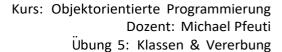
Jeder Drucker soll folgende Methoden haben:

- set_page die Methode nimmt die Seite (Page Klasse) als Argument entgegen. Mit der Methode print page wird dann diese Seite ausgedruckt.
- print page führt den Ausdruck aus und hat keine Argumente. Es muss sichergestellt werden, dass vorher mit set page eine Seite zum Ausdruck angegeben wurde.

1 Klasse: Page

Wir brauchen eine Klasse, welche eine Seite darstellt. Dazu schreiben wir ein Klasse Page, welche nur aus folgenden zwei Attributen besteht.

- text String, welcher der Text der Seite darstellt.
- color String, welcher die Farbe des Textes darstellt. Es sind nur white, black, red, blue, green, yellow gültige Farben.





Die Klasse hat keine Methoden. Der Text und die Farbe werde dem Konstruktor übergeben.

Im Konstruktor soll mit assert geprüft werden, dass eine gültige Farbe übergeben wurde. Dieses Schlüsselwort haben wir noch nicht kennengelernt. Recherchiere was es macht und wie es eingesetzt wird. Was passiert wenn eine falsche Farbe übergeben wird?

```
class Page:
def __init__(self, text, color):
    # Definierte gültige Farben
    valid_colors = ['white', 'black', 'red', 'blue', 'green', 'yellow']
    # Überprüft, ob die übergebene Farbe gültig ist
    assert color in valid_colors, f"{color} is not a valid color"

self.text = text
    self.color = color

# Beispiel zur Verwendung der Page-Klasse
try:
    page = Page("Dies ist ein Beispieltext.", "blue") # Gültige Farbe
print(f"Seite erstellt mit Text: {page.text} in Farbe: {page.color}")

invalid_page = Page("Dies ist der Errorcode.", "purple") # Ungültige Farbe
except AssertionError as error:
print(error)
```

2 Klasse: Schwarz-Weiss Drucker

Dieser Drucker implementiert die Drucker Methoden set page, print page und gibt den Text einfach mit der Python Funktion print aus.

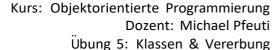
```
class SchwarzWeissDrucker:
def __init__(self):
    self.page = None

def set_page(self, page):
    self.page = page

def print_page(self):
    if self.page is not None:
        print(self.page.text)
else:
    print("Keine Seite zum Drucken festgelegt.")
```

3 Klasse: Farb Drucker

Dieser Drucker implementiert die Drucker Methoden set page, print page und gibt den Text mit der Funktion print color (text, color) aus der Python-Datei print helper.py auf der Kommandozeile aus. Die Funktion soll aus der Python-Datei importiert und verwendet werden.





```
from print_helper import print_color

class FarbDrucker:

def __init__(self):
    self.page = None

def set_page(self, page):
    self.page = page

def print_page(self):
    if self.page is not None:
        print_color(self.page.text, self.page.color)
    else:
    print("Keine Seite zum Drucken festgelegt.")
```

4 Klasse: Bild Drucker

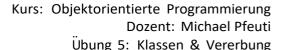
Dieser Drucker implementiert die Drucker Methoden set page, print page und gibt den Text mit der Funktion print on page(letter , color , folder) aus der Python-Datei *print helper.py* als einzelne Bilder im entsprechenden Ordner aus. Die Funktion soll aus der Python-Datei importiert und verwendet werden.

Dieser Funktion muss jeder Buchstabe einzeln übergeben werden. Zudem muss auch der Ordner angegeben werden, in welchem die Bilder gespeichert werden.

```
from print_helper import print_on_page
     class BildDrucker:
         def __init__(self, folder):
             self.page = None
             self.folder = folder
         def set_page(self, page):
             self.page = page
11
         def print page(self):
              if self.page is not None:
12
13
                  for letter in self.page.text:
                      print_on_page(letter, self.page.color, self.folder)
             else:
                  print("Keine Seite zum Drucken festgelegt.")
17
```

5 Testlauf

Es soll wieder eine Python-Datei geschrieben werden, mit welcher die Funktionalität aller Drucker anhand von einem Beispiel demonstriert wird.





```
# Zuerst importieren wir die benötigten Klassen und Funktionen
     from print_helper import print_color, print_on_page
     from PIL import Image, ImageDraw, ImageFont
     import os
     # Definition der Page Klasse
     class Page:
         def __init__(self, text, color):
             valid_colors = ['white', 'black', 'red', 'blue', 'green', 'yellow']
             assert color in valid colors, f"{color} is not a valid color"
             self.text = text
12
             self.color = color
     # Definition der Drucker Klassen
     class SchwarzWeissDrucker:
         def __init__(self):
             self.page = None
         def set_page(self, page):
             self.page = page
         def print_page(self):
             if self.page is not None:
                 print(self.page.text)
             else:
                 print("Keine Seite zum Drucken festgelegt.")
     class FarbDrucker:
         def __init__(self):
             self.page = None
         def set page(self, page):
             self.page = page
         def print_page(self):
             if self.page is not None:
                 print color(self.page.text, self.page.color)
```

Kurs: Objektorientierte Programmierung Dozent: Michael Pfeuti Übung 5: Klassen & Vererbung



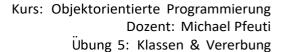
```
print_color(self.page.text, self.page.color)
             else:
                 print("Keine Seite zum Drucken festgelegt.")
     class BildDrucker:
         def init (self, folder):
42
             self.page = None
             self.folder = folder
         def set page(self, page):
             self.page = page
         def print_page(self):
             if self.page is not None:
                 for letter in self.page.text:
                     print_on_page(letter, self.page.color, self.folder)
             else:
                 print("Keine Seite zum Drucken festgelegt.")
     # Erstellen von Beispielseiten und Druckern
     page = Page("Blaue Welt", "blue")
     sw drucker = SchwarzWeissDrucker()
     farb drucker = FarbDrucker()
     bild_drucker = BildDrucker("bilder")
     # Test der Drucker
     sw_drucker.set_page(page)
     sw_drucker.print_page()
     farb_drucker.set_page(page)
     farb_drucker.print_page()
     bild_drucker.set_page(page)
     bild_drucker.print_page()
     print("Testlauf abgeschlossen.")
```

6 Vererbung

Wir haben drei Drucker Klassen. Jede der Klassen hat gewissen Code, der identisch ist. z.B. muss jede Klasse die Methode set page auf genau dieselbe Art implementieren. Somit haben wir dreimal denselben Code.

Diese Methode soll in eine neue Basisklasse Drucker extrahiert werden. Die drei implementierten Drucker sollen nun von dieser Basisklasse erben. Somit kann die Methode set_page in den drei Klassen gelöscht werden, da diese Methode von der Basisklasse vererbt wird.

Es soll wieder mit einem Testlauf überprüft werden, dass auch mit der Basisklasse und der





Vererbung die Funktionalität unverändert ist.

Wir sehen mit Vererbung können wir duplizierten Code vermeiden!

```
# print helper.py muss im gleichen Verzeichnis vorhanden sein
     from print helper import print color, print on page
     # Basisklasse Drucker
     class Drucker:
         def __init__(self):
             self.page = None
         def set_page(self, page):
             self.page = page
12
     # Klasse Page
     class Page:
         def __init__(self, text, color):
             valid_colors = ['white', 'black', 'red', 'blue', 'green', 'yellow']
             assert color in valid_colors, f"{color} is not a valid color"
             self.text = text
             self.color = color
     # Spezifische Drucker-Klassen
     class SchwarzWeissDrucker(Drucker):
         def print_page(self):
             if self.page is not None:
                 print(self.page.text)
             else:
                 print("Keine Seite zum Drucken festgelegt.")
     class FarbDrucker(Drucker):
         def print_page(self):
             if self.page is not None:
                 print_color(self.page.text, self.page.color)
             else:
                 print("Keine Seite zum Drucken festgelegt.")
     class BildDrucker(Drucker):
         def init (self, folder):
             super(). init ()
             self.folder = folder
```





```
def print page(self):
             if self.page is not None:
                 for letter in self.page.text:
                     print_on_page(letter, self.page.color, self.folder)
             else:
                 print("Keine Seite zum Drucken festgelegt.")
     # Testlauf
     if __name__ == "__main__":
         page = Page("Blaue Welt", "blue")
         sw drucker = SchwarzWeissDrucker()
         farb drucker = FarbDrucker()
         bild_drucker = BildDrucker("bilder")
         sw_drucker.set_page(page)
         sw_drucker.print_page()
56
         farb_drucker.set_page(page)
         farb_drucker.print_page()
         bild_drucker.set_page(page)
         bild_drucker.print_page()
         print("Testlauf mit Vererbung abgeschlossen.")
```