

I Bibliothek

Wir wollen eine einfache 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 entsprechend 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)?.

```
1 class Buch:
2     def __init__(self, titel, autor, einband, genre, seiten):
3         self.titel = titel
4         self.autor = autor
5         self.einband = einband
6         self.genre = genre
7         self.seiten = seiten
8         self.geoeffnet = False
9         self.aktuelle_seite = 0
10
11     def __str__(self):
12         zustand = "geöffnet" if self.geoeffnet else "geschlossen"
13         lesefortschritt = f", gelesen bis Seite {self.aktuelle_seite}" if self.geoeffnet and self.aktuelle_seite > 0 else ""
14         return f"Buchtitel: {self.titel}, Autor: {self.autor}, Einband: {self.einband}, Genre: {self.genre}, Zustand: {zustand}{lesefortschritt}."
15
16     def oeffnen(self):
17         if self.geoeffnet:
18             print("Das Buch ist bereits geöffnet.")
19         else:
20             self.geoeffnet = True
21             self.aktuelle_seite = 0
22             print(f"Das Buch '{self.titel}' wurde geöffnet.")
23
24     def schliessen(self):
25         if not self.geoeffnet:
26             print("Das Buch ist bereits geschlossen.")
27         else:
28             self.geoeffnet = False
29             print(f"Das Buch '{self.titel}' wurde geschlossen.")
30
31     def lesen(self):
32         if not self.geoeffnet:
33             print("Das Buch muss zuerst geöffnet werden.")
34         else:
35             if self.aktuelle_seite < len(self.seiten):
36                 print(self.seiten[self.aktuelle_seite])
37                 self.aktuelle_seite += 1
38             if self.aktuelle_seite == len(self.seiten):
39                 print("Das war die letzte Seite. Das Buch wird automatisch geschlossen.")
40                 self.schliessen()
41         else:
42             print("Das Buch wurde bereits bis zum Ende gelesen und wird geschlossen.")
43             self.schliessen()
```

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.

```
1 class Bibliothek:
2     def __init__(self):
3         self.buchbestand = []
4         self.ausgeliehene_buecher = []
5
6     def buch_aufnehmen(self, buch):
7         if buch not in self.buchbestand and buch not in self.ausgeliehene_buecher:
8             self.buchbestand.append(buch)
9             print(f"Buch '{buch.titel}' wurde dem Bestand hinzugefügt.")
10        else:
11            print(f"Buch '{buch.titel}' befindet sich bereits im Bestand.")
12
13    def buch_ausleihen(self, buch):
14        if buch in self.ausgeliehene_buecher:
15            print(f"Buch '{buch.titel}' ist bereits ausgeliehen.")
16        elif buch in self.buchbestand:
17            self.buchbestand.remove(buch)
18            self.ausgeliehene_buecher.append(buch)
19            print(f"Buch '{buch.titel}' wurde ausgeliehen.")
20        else:
21            print(f"Buch '{buch.titel}' befindet sich nicht im Bestand.")
22
23    def buch_zurueckgeben(self, buch):
24        if buch in self.ausgeliehene_buecher:
25            self.ausgeliehene_buecher.remove(buch)
26            self.buchbestand.append(buch)
27            print(f"Buch '{buch.titel}' wurde zurückgegeben und dem Bestand hinzugefügt.")
28        else:
29            print(f"Buch '{buch.titel}' wurde nicht ausgeliehen.")
30
31    def __str__(self):
32        bestands_titel = ", ".join(buch.titel for buch in self.buchbestand)
33        ausgeliehene_titel = ", ".join(buch.titel for buch in self.ausgeliehene_buecher)
34        return f"Bücher im Bestand: {bestands_titel}\nAusgeliehene Bücher: {ausgeliehene_titel}"
35
```

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.

```
1 # Zuerst müssen wir die Klassen Buch und Bibliothek definieren, wie oben beschrieben.
2
3 # Klasse Buch
4 class Buch:
5     def __init__(self, titel, autor, einband, genre, seiten):
6         self.titel = titel
7         self.autor = autor
8         self.einband = einband
9         self.genre = genre
10        self.seiten = seiten
11        self.geoeffnet = False
12        self.aktuelle_seite = 0
13
14    def __str__(self):
15        zustand = "geöffnet" if self.geoeffnet else "geschlossen"
16        lesefortschritt = f", gelesen bis Seite {self.aktuelle_seite}" if self.geoeffnet and self.aktuelle_seite > 0 else ""
17        return f"Buchtitel: {self.titel}, Autor: {self.autor}, Einband: {self.einband}, Genre: {self.genre}, Zustand: {zustand}{lesefortschritt}."
18
19    def oeffnen(self):
20        if self.geoeffnet:
21            print("Das Buch ist bereits geöffnet.")
22        else:
23            self.geoeffnet = True
24            self.aktuelle_seite = 0
25            print(f"Das Buch '{self.titel}' wurde geöffnet.")
26
27    def schliessen(self):
28        if not self.geoeffnet:
29            print("Das Buch ist bereits geschlossen.")
30        else:
31            self.geoeffnet = False
32            print(f"Das Buch '{self.titel}' wurde geschlossen.")
33
34    def lesen(self):
35        if not self.geoeffnet:
36            print("Das Buch muss zuerst geöffnet werden.")
37        else:
38            if self.aktuelle_seite < len(self.seiten):
39                print(self.seiten[self.aktuelle_seite])
40                self.aktuelle_seite += 1
41            if self.aktuelle_seite == len(self.seiten):
42                print("Das war die letzte Seite. Das Buch wird automatisch geschlossen.")
43                self.schliessen()
44
45        else:
46            print("Sie haben bereits das Ende des Buches erreicht.")
47            self.schliessen()
48
49 # Klasse Bibliothek
50 class Bibliothek:
51     def __init__(self):
52         self.buchbestand = []
53         self.ausgeliehene_buecher = []
54
55     def buch_aufnehmen(self, buch):
56         self.buchbestand.append(buch)
57         print(f"Buch '{buch.titel}' wurde dem Bestand hinzugefügt.")
58
59     def buch_ausleihen(self, buch):
60         if buch in self.ausgeliehene_buecher:
61             print(f"Buch '{buch.titel}' ist bereits ausgeliehen.")
62         elif buch in self.buchbestand:
63             self.buchbestand.remove(buch)
64             self.ausgeliehene_buecher.append(buch)
65             print(f"Buch '{buch.titel}' wurde ausgeliehen.")
66         else:
67             print(f"Buch '{buch.titel}' ist nicht im Bestand.")
68
69     def buch_zurueckgeben(self, buch):
70         if buch in self.ausgeliehene_buecher:
71             self.ausgeliehene_buecher.remove(buch)
72             self.buchbestand.append(buch)
73             print(f"Buch '{buch.titel}' wurde zurückgegeben.")
74         else:
75             print(f"Buch '{buch.titel}' war nicht ausgeliehen.")
76
77     def __str__(self):
78         bestands_titel = ", ".join(buch.titel for buch in self.buchbestand)
79         ausgeliehene_titel = ", ".join(buch.titel for buch in self.ausgeliehene_buecher)
80         return f"Bücher im Bestand: {bestands_titel}\nAusgeliehene Bücher: {ausgeliehene_titel}"
81
82 # Testlauf
83 if __name__ == "__main__":
84     # Erstellen einiger Bücher
85     buch1 = Buch("Python Programmierung", "Max Mustermann", "Hardcover", "Informatik", ["Seite 1 Inhalt", "Seite 2 Inhalt"])
86     buch2 = Buch("Das Leben des Pi", "Yann Martel", "Paperback", "Abenteuer", ["Seite 1 Inhalt", "Seite 2 Inhalt"])
87     buch3 = Buch("Der Große Gatsby", "F. Scott Fitzgerald", "Hardcover", "Klassiker", ["Seite 1 Inhalt", "Seite 2 Inhalt"])
```

```
87
88     # Erstellen der Bibliothek und Hinzufügen der Bücher
89     bibliothek = Bibliothek()
90     bibliothek.buch_aufnehmen(buch1)
91     bibliothek.buch_aufnehmen(buch2)
92     bibliothek.buch_aufnehmen(buch3)
93
94     # Ausleihen und Zurückgeben von Büchern
95     bibliothek.buch_ausleihen(buch1)
96     bibliothek.buch_ausleihen(buch1) # Versuch, ein bereits ausgeliehenes Buch auszuleihen
97     bibliothek.buch_zurueckgeben(buch1)
98     bibliothek.buch_zurueckgeben(buch1) # Versuch, ein bereits zurückgegebenes Buch zurückzugeben
99
100    # Demonstration des Lesens eines Buches
101    buch1.oeffnen()
102    buch1.lesen()
103    buch1.lesen()
104    buch1.schliessen()
105    buch1.lesen() # Versuch zu lesen, nachdem das Buch geschlossen wurde
106
107    # Anzeigen des aktuellen Bibliotheksbestands und der ausgeliehenen Bücher
108    print(bibliothek)
109
```

II Drucker

In dieser Übung 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?

```
1 class Page:
2     def __init__(self, text, color):
3         # Definierte gültige Farben
4         valid_colors = ['white', 'black', 'red', 'blue', 'green', 'yellow']
5         # Überprüft, ob die übergebene Farbe gültig ist
6         assert color in valid_colors, f"{color} is not a valid color"
7
8         self.text = text
9         self.color = color
10
11 # Beispiel zur Verwendung der Page-Klasse
12 try:
13     page = Page("Dies ist ein Beispieltext.", "blue") # Gültige Farbe
14     print(f"Seite erstellt mit Text: {page.text} in Farbe: {page.color}")
15
16     invalid_page = Page("Dies ist der Errorcode.", "purple") # Ungültige Farbe
17 except AssertionError as error:
18     print(error)
19
```

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.

```
1 class SchwarzWeissDrucker:
2     def __init__(self):
3         self.page = None
4
5     def set_page(self, page):
6         self.page = page
7
8     def print_page(self):
9         if self.page is not None:
10            print(self.page.text)
11        else:
12            print("Keine Seite zum Drucken festgelegt.")
13
```

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.

```
1  from print_helper import print_color
2
3  class FarbDrucker:
4      def __init__(self):
5          self.page = None
6
7      def set_page(self, page):
8          self.page = page
9
10     def print_page(self):
11         if self.page is not None:
12             print_color(self.page.text, self.page.color)
13         else:
14             print("Keine Seite zum Drucken festgelegt.")
15
```

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.

```
1  from print_helper import print_on_page
2
3  class BildDrucker:
4      def __init__(self, folder):
5          self.page = None
6          self.folder = folder
7
8      def set_page(self, page):
9          self.page = page
10
11     def print_page(self):
12         if self.page is not None:
13             for letter in self.page.text:
14                 print_on_page(letter, self.page.color, self.folder)
15         else:
16             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.


```
1  # Zuerst importieren wir die benötigten Klassen und Funktionen
2  from print_helper import print_color, print_on_page
3  from PIL import Image, ImageDraw, ImageFont
4  import os
5
6  # Definition der Page Klasse
7  class Page:
8      def __init__(self, text, color):
9          valid_colors = ['white', 'black', 'red', 'blue', 'green', 'yellow']
10         assert color in valid_colors, f"{color} is not a valid color"
11         self.text = text
12         self.color = color
13
14     # Definition der Drucker Klassen
15     class SchwarzWeissDrucker:
16         def __init__(self):
17             self.page = None
18
19         def set_page(self, page):
20             self.page = page
21
22         def print_page(self):
23             if self.page is not None:
24                 print(self.page.text)
25             else:
26                 print("Keine Seite zum Drucken festgelegt.")
27
28     class FarbDrucker:
29         def __init__(self):
30             self.page = None
31
32         def set_page(self, page):
33             self.page = page
34
35         def print_page(self):
36             if self.page is not None:
37                 print_color(self.page.text, self.page.color)
```

```
37         print_color(self.page.text, self.page.color)
38     else:
39         print("Keine Seite zum Drucken festgelegt.")
40
41     class BildDrucker:
42     def __init__(self, folder):
43         self.page = None
44         self.folder = folder
45
46     def set_page(self, page):
47         self.page = page
48
49     def print_page(self):
50         if self.page is not None:
51             for letter in self.page.text:
52                 print_on_page(letter, self.page.color, self.folder)
53         else:
54             print("Keine Seite zum Drucken festgelegt.")
55
56     # Erstellen von Beispielseiten und Druckern
57     page = Page("Blaue Welt", "blue")
58     sw_drucker = SchwarzWeissDrucker()
59     farb_drucker = FarbDrucker()
60     bild_drucker = BildDrucker("bilder")
61
62     # Test der Drucker
63     sw_drucker.set_page(page)
64     sw_drucker.print_page()
65
66     farb_drucker.set_page(page)
67     farb_drucker.print_page()
68
69     bild_drucker.set_page(page)
70     bild_drucker.print_page()
71
72     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!

```
1  # print_helper.py muss im gleichen Verzeichnis vorhanden sein
2  from print_helper import print_color, print_on_page
3
4  # Basisklasse Drucker
5  class Drucker:
6      def __init__(self):
7          self.page = None
8
9      def set_page(self, page):
10         self.page = page
11
12  # Klasse Page
13  class Page:
14      def __init__(self, text, color):
15         valid_colors = ['white', 'black', 'red', 'blue', 'green', 'yellow']
16         assert color in valid_colors, f"{color} is not a valid color"
17         self.text = text
18         self.color = color
19
20  # Spezifische Drucker-Klassen
21  class SchwarzWeissDrucker(Drucker):
22      def print_page(self):
23         if self.page is not None:
24             print(self.page.text)
25         else:
26             print("Keine Seite zum Drucken festgelegt.")
27
28  class FarbDrucker(Drucker):
29      def print_page(self):
30         if self.page is not None:
31             print_color(self.page.text, self.page.color)
32         else:
33             print("Keine Seite zum Drucken festgelegt.")
34
35  class BildDrucker(Drucker):
36      def __init__(self, folder):
37         super().__init__()
38         self.folder = folder
```

```
39
40     def print_page(self):
41         if self.page is not None:
42             for letter in self.page.text:
43                 print_on_page(letter, self.page.color, self.folder)
44         else:
45             print("Keine Seite zum Drucken festgelegt.")
46
47     # Testlauf
48     if __name__ == "__main__":
49         page = Page("Blaue Welt", "blue")
50         sw_drucker = SchwarzWeissDrucker()
51         farb_drucker = FarbDrucker()
52         bild_drucker = BildDrucker("bilder")
53
54         sw_drucker.set_page(page)
55         sw_drucker.print_page()
56
57         farb_drucker.set_page(page)
58         farb_drucker.print_page()
59
60         bild_drucker.set_page(page)
61         bild_drucker.print_page()
62
63         print("Testlauf mit Vererbung abgeschlossen.")
```