2025/10/17 16:47 1/5 LU07.L01 - Bibliothek

LU07.L01 - Bibliothek

customer.py

```
""" Provides the class Customer for the Library application."""
class Customer:
    Defines a customer of our library who,
    * borrows [ borrow_book_by_title() ] or
    * returns [ bring_back_book() ]
    a book.
    Each customer may only borrow one book at a time.
    Attributes
    name: str
        The full name of this customer.
    reminded: bool (Default: false)
        Has this customer been reminded to return an overdue book.
    librarian: Librarian
        A reference to a librarian of our library.
    book: Book (Default: None)
        The book currently borrowed by this customer. None=no book
borrowed
    0.00
    def __init__(self, name, librarian, library):
        Creates a customer object, sets the reference to the librarian
        and registers the client with the library.
        :param name: Full name of this customer.
        :param librarian: A reference to the librarian.
        :param library: A reference to the library.
        self. name = name
        self. reminded = False
        self. librarian = librarian
        self. book = None
        library.add_customer(self)
    def __str__(self):
        Gibt den Namen des Kunden aus.
        return f'Kunde: {self.name}'
    def borrow book by title(self, title):
```

```
0.00
        Borrows a book identified by the title from the librarian.
        If the book is available, the reference to the book will be
set.
        Otherwise, an error message is printed.
        :param title: The title of the book.
        self._book = self._librarian.lend_book_by_title(title)
        if self. book is not None:
            print(f'{self.name} hat das Buch "{self.book.title}"
erhalten.')
    def bring back book(self):
        Returns the book to the librarian.
        The reference to the book will be set to None.
        print(f'{self.name} hat das Buch "{self.book.title}"
zurückgebracht')
        self. librarian.take back book(self. book)
        self. book = None
    @property
    def name(self):
        Gets the name of this customer.
        :return: Name des Kunden
        return self. name
    @property
    def book(self):
        Gets the name of the borrowed book.
        :return: Referenz zum ausgeliehenen Buch
        return self._book
    @property
    def reminded(self):
        Gets the status of the reminder.
        :return: Status der Mahnung true/false
        return self. reminded
    @reminded.setter
    def reminded(self, value):
```

https://wiki.bzz.ch/ Printed on 2025/10/17 16:47

2025/10/17 16:47 3/5 LU07.L01 - Bibliothek

```
Sets the status of the reminder.

self._reminded = value
```

librarian.py

```
""" Provides the class Librarian for the Library application."""
from book import Book
from library import Library
class Librarian:
    Defines the librarian, who:
    * buys new books, buy_new_book(...)
    * lends books to the customer, lend book by title(...)
    * takes back the books, take_back_book(...)
    * remindes overdue customers, remind customer(...)
    * removes books from the library, remove_book(...)
    Attributes
    _ _ _ _ _ _ _ _ _
    name: str
        The full name of the librarian.
    library: Library
        A reference to the library this librarian works for.
    .....
    def __init__(self, name, library):
        Creates the librarian-object with his name
        and a reference to the library.
        :param name: The full name of this librarian.
        :param library: A reference to the library.
        self. name = name
        self. library = library
    def buy new book(self, title, isbn):
        Creates a new book with the title and ISBN-number to the
library.
        Adds this book to the library and saves the location (shelf).
        :param title: The title of the new book.
        :param isbn: The ISBN-number of the new book.
        book = Book(title, isbn)
```

```
book.location = self._library.add_book(book)
   def lend book by title(self, title):
        Lends the book with the specified title to a customer.
        If no book with this title is available, it prints a message.
        :param title: The title of the requested book.
        :return: book-object or None=not dound
        book = self. library.search book by title(title)
        if book is None:
            print('Das angefragte Buch ist nicht vorhanden')
        else:
            location = book.location
            book = self. library.lend book(location)
        return book
   def take back book(self, borrowed book):
        Takes back a book and returns it to the library.
        :param borrowed book: The book given back by the customer.
        self. library.reshelve book(borrowed book)
   def remove book(self, title):
        Removes the book with the specified title from the library.
        :param title: The title of the book to be removed
        :raise: LookupError when there is no book with the specified
title.
        print(f'\n---\nentferne Buch "{title}"')
        book = self._library.search_book_by_title(title)
        self._library.remove_book(book)
   def remind customer(self, name):
        Reminds a customer, that a book is overdue.
        :param name: The name of the customer to be reminded.
        customer = self._library.search_customer(name)
        customer.reminded = True
        print(f'Erinnerung für {name}: Dein Buch ist überfällig')
```

https://wiki.bzz.ch/ Printed on 2025/10/17 16:47

From:

https://wiki.bzz.ch/ - BZZ - Modulwiki

Permanent link:

https://wiki.bzz.ch/modul/m320_2024/learningunits/lu07/loesungen/bibliothek

Last update: 2024/09/18 06:11

