ОТЧЕТ

Лабораторная работа №4

Объектно-ориентированное программирование на языке Python. Инкапсуляция данных, наследование, полиморфизм. Создание экземпляров объектов

Выполнила: Балабанова Н.И.

Группа: ПИН-31Д

Москва 2023

Задание. Реализовать класс-родитель: «Книга».

Спроектировать несколько классов, используя механизм наследования.

Реализовать два метода:

Функция-метод №1: вычислить среднюю стоимость одной страницы.

Функция-метод №2: увеличить цену книги в два раза, если название начинается со слова «Программирование».

Данная лабораторная работа состоит из двух основных файлов: `book.py` и `lab_4_1.py`

В файле `book.py` объявлена реализация класса "Книга" и его наследников:

```
class Book:

    def __init__(self,title,author,publisher,pages,cost):
        self.__title = title
        self.__author = author
        self.__publisher = publisher
        self.__pages = pages
        self.__cost = cost

    def costOfPage(self):
        return self.getCost() / self.__pages

    def getCost(self):
        if str(self.__title).lower().__contains__("программировани"):
            return self.__cost * 2

        return self.__cost

    def __str__(self):
        return "<{3}>:\n\ttitle: \t\t{0}\n\tauthor: \t{4}\n\tpublisher:
\t{5}\n\tpages: \t\t{1}\n\tcost: \t\t{2}".format(self.__title, self.__pages,
self.__cost, type(self), self.__author, self.__publisher)


class AudioBook(Book):
    def __init__(self,title,author,publisher,pages,cost,read):
        Book.__init__(self,title,author,publisher,pages,cost)
        self.__read = read

    def __str__(self):
        return super().__str__() + "\n\tread: \t\t{}".format(self.__read)

class ForeignBook(Book):
    def __init__(self,title,author,publisher,pages,cost,interpreter):
        Book.__init__(self,title,author,publisher,pages,cost)
        self.__interpreter = interpreter

    def __str__(self):
        return super().__str__() + "\n\tinterpreter: \t{}".format(self.__interpreter)

class ForeignAudioBook(AudioBook,ForeignBook):
    def __init__(self,title,author,publisher,pages,cost,read,interpreter):
        ForeignBook.__init__(self,title,author,publisher,pages,cost,interpreter)
        AudioBook.__init__(self,title,author,publisher,pages,cost,read)
```

В файле `lab_4_1.py` представлена демонстрация работы наследования:

```
import book

b = book.Book("Колыбель для кошки", "Курт Воннегут", "АСТ", 250, 600)
ab = book.AudioBook("Колыбель для кошки", "Курт Воннегут", "АСТ", 250, 600, "Д.
Пучков")
fb = book.ForeignBook("Колыбель для кошки", "Курт Воннегут", "АСТ", 250, 600, "Р.Райт-
Ковалева")
fab = book.ForeignAudioBook("Колыбель для кошки", "Курт Воннегут", "АСТ", 250, 600,
"Д. Пучков", "Р.Райт-Ковалева")


fb2 = book.ForeignBook("Исскуство программирования", "Дональд Кнут", "Диалектика
(Вильямс)", 960, 4761, "С. Г. Тригуб, Ю. Г. Гордиенко, И. В. Красиков и др.")


def printClass(book):
    print(book,"\n")
    print("\tgetCost(): \t", book.getCost())
    print("\tcostOfPage(): \t", book.costOfPage(),"\n")


printClass(fb)
printClass(fb2)
printClass(fab)
```

printClass	**__str__**
fb (ForeignBook)	<<class 'book.ForeignBook'>>: title: Колыбель для кошки author: Курт Воннегут publisher: АСТ pages: 250 cost: 600 interpreter: Р.Райт-Ковалева getCost(): 600 costOfPage(): 2.4
fb2 (ForeignBook)	<<class 'book.ForeignBook'>>: title: Исскуство программирования author: Дональд Кнут publisher: Диалектика (Вильямс) pages: 960 cost: 4761 interpreter: С. Г. Тригуб, Ю. Г. Гордиенко, И. В. Красиков и др. getCost(): 9522 costOfPage(): 9.91875

printClass	__str__
fab (ForeignAudioBook)	<<class 'book.ForeignAudioBook'>>: title: Колыбель для кошки author: Курт Воннегут publisher: АСТ pages: 250 cost: 600 interpreter: Р.Райт-Ковалева read: Д. Пучков getCost(): 600 costOfPage(): 2.4