**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Информатика»**

**Тема: Парадигмы программирования**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Гребенюк В.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Целью работы является освоение работы объектно-ориентированной парадигмой программирования в Python.

## Задание

Вариант 4

Базовый класс — печатное издание Edition:

class Edition:

Поля объекта класса Edition:

название (строка)

цена (в руб., целое положительное число)

возрастное ограничение (в годах, целое положительное число)

стиль(значение может быть одной из строк: c (color), b (black))

При создании экземпляра класса Edition необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

Книга - Book:

class Book: #Наследуется от класса Edition

Поля объекта класс Book:

название (строка)

цена (в руб., целое положительное число)

возрастное ограничение (в годах, целое положительное число)

стиль(значение может быть одной из строк: c (color), b (black))

автор (фамилия, в виде строки)

твердый переплет (значениями могут быть или True, или False)

количество страниц (целое положительное число)

При создании экземпляра класса Book необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

В данном классе необходимо реализовать следующие методы:

Метод \_\_str\_\_():

Преобразование к строке вида: Book: название <название>, цена <цена>, возрастное ограничение <возрастное ограничение>, стиль <стиль>, автор <автор>, твердый переплет <твердый переплет>, количество страниц <количество страниц>.

Метод \_\_eq\_\_():

Метод возвращает True, если два объекта класса равны и False иначе. Два объекта типа Book равны, если равны их название и автор.

Газета - Newspaper:

class Newspaper: #Наследуется от класса Edition

Поля объекта класс Newspaper:

название (строка)

цена (в руб., целое положительное число)

возрастное ограничение (в годах, целое положительное число)

стиль(значение может быть одной из строк: c (color), b (black))

интернет издание (значениями могут быть или True, или False)

страна (строка)

периодичность (период выпуска газеты в днях, целое положительное число)

При создании экземпляра класса Newspaper необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

В данном классе необходимо реализовать следующие методы:

Метод \_\_str\_\_():

Преобразование к строке вида: Newspaper: название <название>, цена <цена>, возрастное ограничение <возрастное ограничение>, стиль <стиль>, интернет издание <интернет издание>, страна <страна>, периодичность <периодичность>.

Метод \_\_eq\_\_():

Метод возвращает True, если два объекта класса равны и False иначе. Два объекта типа Newspaper равны, если равны их название и страна.

Необходимо определить список list для работы с печатным изданием:

Книги:

class BookList – список книг - наследуется от класса list.

Конструктор:

Вызвать конструктор базового класса.

Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

Метод append(p\_object): Переопределение метода append() списка. В случае, если p\_object - книга, элемент добавляется в список, иначе выбрасывается исключение TypeError с текстом: Invalid type <тип\_объекта p\_object> (результат вызова функции type)

Метод total\_pages(): Метод возвращает сумму всех страниц всех имеющихся книг.

Метод print\_count(): Вывести количество книг.

Газеты:

class NewspaperList – список газет - наследуется от класса list.

Конструктор:

Вызвать конструктор базового класса.

Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

Метод extend(iterable): Переопределение метода extend() списка. В случае, если элемент iterable - объект класса Newspaper, этот элемент добавляется в список, иначе не добавляется.

Метод print\_age(): Вывести самое низкое возрастное ограничение среди всех газет.

Метод print\_total\_price(): Посчитать и вывести общую цену всех газет.

В отчете укажите:

Изображение иерархии описанных вами классов.

Методы, которые вы переопределили (в том числе методы класса object).

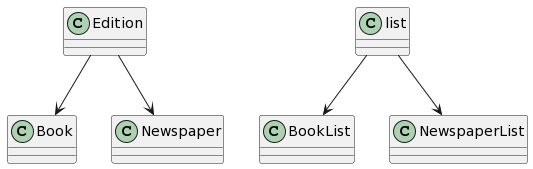
В каких случаях будут использованы методы \_\_str\_\_() и \_\_eq\_\_().

Будут ли работать переопределенные методы класса list для BookList и NewspaperList? Объясните почему и приведите примеры.

## Выполнение работы

Создан код в соответствии с заданием.

1. Иерархия классов:

Рисунок 1 – Иерархия классов

2. Были переопределенны методы: *\_\_str\_\_(), \_\_eq\_\_()*, *\_\_init\_\_(), append(), extend()*

3. Метод *\_\_str\_\_()* будет вызываться при касте в тип строки (*str*)

Метод *\_\_eq\_\_()* будетвызываться при сравнении (*==*)

4. Переопределённые методы классв *list* для *BookList* и *NewspaperList* будут работать. Потому что переопределённые методы класса *list* всё равно вызываются через метод *super()*, который возвращает унаследованный рожительский класс со всеми его методами.

Например при вызове *super().extend()* в классе *NewspaperList* вызывается метод *extend()* родительского класса, а не рекурсивный вызов тогоже самого метода.

Та же ситуация происходит с *super().append()* в классе *BookList* c методом *append()* родительского класса.

Разработанный программный код см. в приложении А.

## Выводы

Объектно-ориентированная парадигма в Python позволяет создавать структуры данных – классы с собственными методами и полями, с возможностью их наследования и использования в дочерних классах.

# Приложение А Исходный код программы

Название файла: main.py

class Edition:

def \_\_init\_\_(self, name: str, price: int, age\_limit: int, style: str) -> None:

if (

type(name) is not str

or type(price) is not int

or type(age\_limit) is not int

or type(style) is not str

) or ( # sanity check

price < 1

or age\_limit < 1

or style not in ["c", "b"]

or not name

or not style

):

raise ValueError("Invalid value")

self.name = name

self.price = price

self.age\_limit = age\_limit

self.style = style

class Book(Edition):

def \_\_init\_\_(

self,

name: str,

price: int,

age\_limit: int,

style: str,

author: str,

hardcover: bool,

pages: int,

) -> None:

super().\_\_init\_\_(name, price, age\_limit, style)

if (

type(author) is not str

or type(hardcover) is not bool

or type(pages) is not int

) or ( # sanity check

not author or pages < 1

):

raise ValueError("Invalid value")

self.author = author

self.hardcover = hardcover

self.pages = pages

def \_\_str\_\_(self):

return f"Book: название {self.name}, цена {self.price}, возрастное ограничение {self.age\_limit}, стиль {self.style}, автор {self.author}, твердый переплет {self.hardcover}, количество страниц {self.pages}."

def \_\_eq\_\_(self, other):

return (

type(other) is Book

and other.author == self.author

and other.name == self.name

)

class Newspaper(Edition):

def \_\_init\_\_(

self,

name: str,

price: int,

age\_limit: int,

style: str,

online\_edition: bool,

country: str,

frequency: int,

) -> None:

super().\_\_init\_\_(name, price, age\_limit, style)

if (

type(online\_edition) is not bool

or type(country) is not str

or type(frequency) is not int

) or ( # sanity check

not country or frequency < 1

):

raise ValueError("Invalid value")

self.online\_edition = online\_edition

self.country = country

self.frequency = frequency

def \_\_str\_\_(self):

return f"Newspaper: название {self.name}, цена {self.price}, возрастное ограничение {self.age\_limit}, стиль {self.style}, интернет издание {self.online\_edition}, страна {self.country}, периодичность {self.frequency}."

def \_\_eq\_\_(self, other):

return (

type(other) is Newspaper

and other.name == self.name

and other.country == self.country

)

class BookList(list):

def \_\_init\_\_(self, name) -> None:

super().\_\_init\_\_()

self.name = name

def append(self, p\_object):

if type(p\_object) is not Book:

raise TypeError(f"Invalid type {type(p\_object)}")

super().append(p\_object)

def total\_pages(self):

return sum(map(lambda x: x.pages, self))

def print\_count(self):

print(len(self))

class NewspaperList(list):

def \_\_init\_\_(self, name) -> None:

super().\_\_init\_\_()

self.name = name

def extend(self, iterable):

super().extend(filter(lambda x: type(x) is Newspaper, iterable))

def print\_age(self):

print(min(map(lambda x: x.age\_limit, self)))

def print\_total\_price(self):

print(sum(map(lambda x: x.price, self)))