МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Информатика»
Тема: Парадигмы программирования

Студент гр. 3343 _____ Жучков О.Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Целью работы является изучение основ объектно-ориентированного программирования. На языке Python необходимо изучить работу с классами, создание методов для классов, наследование, переопределение методов.

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.

Выполнение работы

Класс Edition содержит такие поля, как название, цена, возрастное ограничение и стиль печатного издания. От него наследуются классы Book и Newspaper. Класс Book имеет поля: автор, количество страниц, жесткость переплета. Класс Newspaper имеет поля: интернет издание газеты, страна, периодичность выпуска. При создании объектов данных классов происходит проверка переданных в конструктор аргументов, если значение для некоторого поля не соответствует требованиям, то выводится исключение ValueError. Класс Book имеет методы для вывода информации об объекте в виде строки и для сравнения двух книг по автору и названию. Класс Newspaper имеет методы для вывода информации об объекте, сравнения двух газет по названию и стране.

Методы __str__ и __eq__, определенные в классах Book и Newspaper, являются специальными методами. Метод __str__ возвращает строковое представление объекта и вызывается при использовании таких функций, как print() или str(). Метод __eq__ используется для определения равенства двух объектов, при вызове выражения a == b для объекта a вызывается метод __eq__ с объектом b в качестве аргумента.

Класс BookList наследуется от list. Переопределён метод append(), в список можно добавить только книгу. Включены методы для счета количества страниц во всех книгах списка и для вывода количества книг в списке. Класс NewspaperList также наследуется от list. Переопределён метод extend(): если элемент итерируемого объекта – газета, то он добавляется в список. Есть методы для вывода наименьшего возрастного ограничения из газет списка, для вывода общей цены всех газет.

В методах append и extend, переопределённых в классах BookList и NewspaperList, вызываются соответствующие методы класса list с помощью функции super(), что обеспечивает их корректную работу.

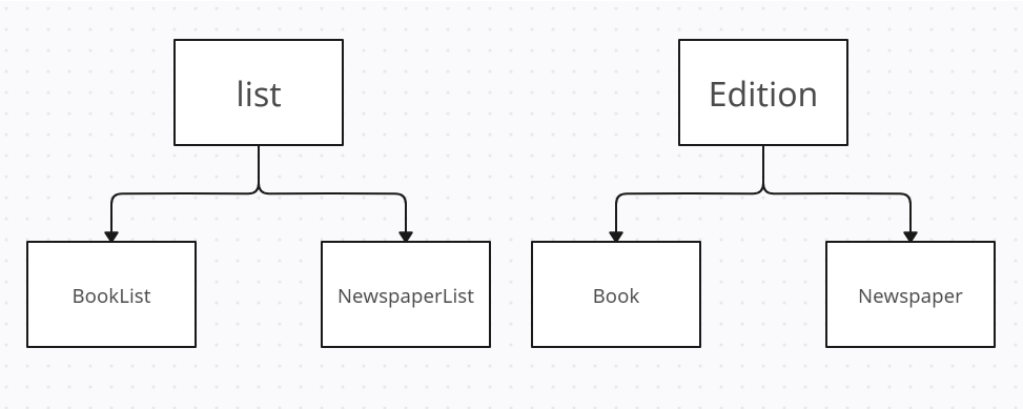

Рисунок 1 – Иерархия созданных классов

Выводы

В ходе выполнения работы была изучены основы объектно-ориентированного программирования на языке Python. Изучены создание классов, создание функций и методов для классов, работа с классами, наследование классов, переопределение методов.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class Edition:
    def __init__(self,name,price,age_limit,style):
        if not isinstance(name, str):
            raise ValueError("Invalid value")
        if not isinstance(price, int):
            raise ValueError("Invalid value")
        if price <= 0:
            raise ValueError("Invalid value")
        if not isinstance(age_limit,int):
            raise ValueError("Invalid value")
        if age_limit <= 0:
            raise ValueError("Invalid value")
        if not isinstance(style,str):
            raise ValueError("Invalid value")
        if style not in ["b", "c"]:
            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, price, age_limit, style, author,
hardcover, pages):
        super().__init__(name,price,age_limit,style)
        if not isinstance(author, str):
            raise ValueError("Invalid value")
        if not isinstance(hardcover, bool):
            raise ValueError("Invalid value")
        if not isinstance(pages, int):
            raise ValueError("Invalid value")
        if pages <= 0:
            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 self.name == other.name and self.author ==
other.author


class Newspaper(Edition):
    def __init__(self, name, price, age_limit, style, online_edition,
country, frequency):
```

```
            super().__init__(name,price,age_limit,style)
            if not isinstance(online_edition, bool):
                raise ValueError("Invalid value")
            if not isinstance(country, str):
                raise ValueError("Invalid value")
            if not isinstance(frequency, int):
                raise ValueError("Invalid value")
            if frequency <= 0:
                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 self.name == other.name and self.country == other.country


    class BookList(list):
        def __init__(self, name):
            super().__init__()
            self.name = name

        def append(self, p_object):
            if isinstance(p_object, Book):
                super().append(p_object)
            else:
                raise TypeError(f"Invalid type {type(p_object)}")

        def total_pages(self):
            cnt = 0
            for book in self:
                cnt += book.pages
            return cnt

        def print_count(self):
            print(len(self))


    class NewspaperList(list):
        def __init__(self, name):
            super().__init__()
            self.name = name

        def extend(self, iterable):
            for paper in iterable:
                if isinstance(paper, Newspaper):
                    super().append(paper)

        def print_age(self):
            age_limits = [paper.age_limit for paper in self]
            min_age = min(age_limits)
```

```
    print(min_age)

def print_total_price(self):
    cost = 0
    for paper in self:
        cost += paper.price
    print(cost)
```