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:

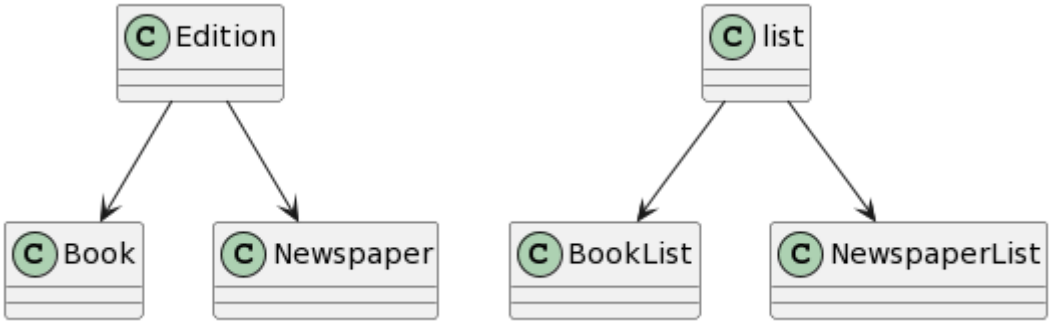

Рисунок 1 – Иерархия классов

2. Были переопределенны методы: *__str__()*, *__eq__()*, *__init__()*, *append()*, *extend()*

3. Метод *__str__()* будет вызываться при касте в тип строки (*str*)

Метод *__eq__()* будет вызываться 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* с 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)))
```