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

ОТЧЕТ

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

Студент гр. 3343 _____ Малиновский А.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Узнать основные принципы работ с классами в языке программирования Python используя объектно-ориентированное программирование.

Задание

Базовый класс - ***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***: (Наследуется от класса *Edition*)

Поля объекта класса *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:

1) 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>.

2) Метод *__eq__()*:

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

Поля объекта класса *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:

1) 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>.

2) 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 фигурами:

Список книг *(class BookList)*:

Конструктор:

- Вызвать конструктор базового класса.
- Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

1) Метод *append(p_object):*

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)

2) Метод *total_pages():*

Метод возвращает сумму всех страниц всех имеющихся книг.

Список газет (*class NewspaperList*):

Конструктор:

- Вызвать конструктор базового класса.
- Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

1) Метод *extend(iterable):*

Переопределение метода extend() списка. В качестве аргумента передается итерируемый объект iterable, в случае, если элемент iterable - объект класса Newspaper, этот элемент добавляется в список, иначе не добавляется.

2) Метод *print_age():*

   Вывести самое низкое возрастное ограничение среди всех газет.

3) Метод *print_total_price():*

   Посчитать и вывести общую цену всех газет.

Выполнение работы

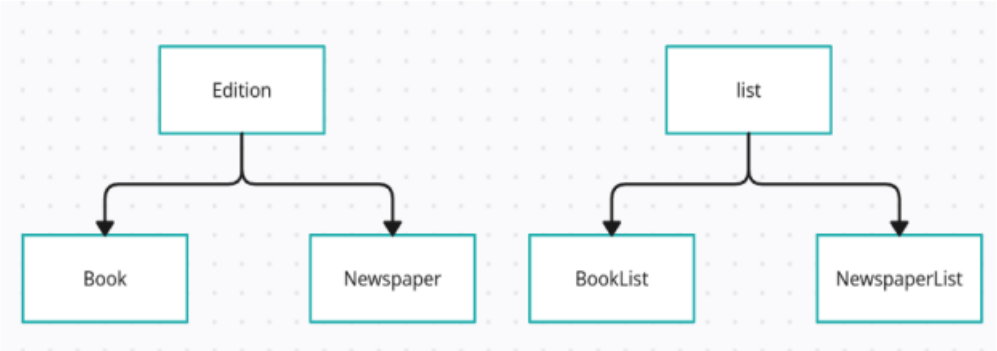

Рисунок 1 - Иерархия созданных классов

Методы разработанных классов

1) Методы классов-наследников Edition

__init__ – конструктор класса, принимающий следующие параметры: *name, price, age_limit, style* – соотвествующие имени, цене, возрастному ограничению и цветовому стилю издания; *author, hardcover, pages* – дополнительные поля класса Book: автор, тип переплёта, количество страниц; *Newspaper* же принимает дополнительные параметры *online_edition, country, frequency* – тип издания, страна и частота выпусков газеты. При инициализации идёт проверка всех типов данных на соответствие требованиям задания, в случае несовпадения поднимается исключение *ValueError*.

__str__ – метод, возвращающий строку данных о книге или газете, в соответствии с требованиями текста задания

__eq__ – метод, переопределяющий оператор сравнения между объектами типов *book* и *newspaper*, на основе сравнения значений заданных полей (т. е. например книги считаются равными, в случае равенства имени автора и названия)

2) Методы классов-наследников List

__init__ – конструктор класса, принимающий параметр *name*, с инициализацией соотвествующего поля и последующим вызовом родительского конструктора

append(p_object) – метод, реализующий добавление одного элемента в *Booklist,* с предварительной проверкой типа на соответствие типу Book с поднятием исключения *TypeError* в случае несоответствия.

extend(iterable) – метод, аналогичный списковому методу *extend* метод класса *NewspaperList*, с игнорированием иных типов кроме *Newspaper*.

total_pages() – метод класса *BookList* возвращающий суммы числа всех страниц входящих в список

print_count() – метод класса *BookList*, выводящий в консоль количество элеметов списка

print_age() – метод класса *NewspaperList*, выводящий минимальное возрастное ограничение, содержащееся в списке

print_total_price() – метод класса *NewspaperList*, возвращающий суммарную цену всех газет.

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

Да, будут работать, потому что они используют функцию *super()* для вызова соотвествующих методов родительского класса, поэтому дополнительные проверки не мешают основной работе метода.

Примером может послужить метод *append* у *BookList* – в коде программы реализован лишь один дополнительный *if* для установления соответсвия типу и вызову ошибки в случае несовпадения, далее тут же

следует функция *super*(), через которую фактически осуществляется работа метода *append* как у *list*.

Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	a = Book("Book", 100, 18, 'c', "A", True, 100) b = Newspaper("Newspaper", 100, 18, 'c', True, "B", 1) c = Newspaper("Newspaper", 100, 16, 'c', False, "B", 2) d = 10 e = '1'	[Book: название Book, цена 100, возрастное ограничение 18, стиль c, автор A, твердый переплет True, количество страниц 100] [Newspaper: название Newspaper, цена 100, возрастное ограничение 18, стиль c, интернет издание True, страна B, периодичность 1, Newspaper: название Newspaper, цена 100, возрастное ограничение 16, стиль c, интернет издание False, страна B, периодичность 2]	Программа сработала корректно.

Выводы

Было изучены основные парадигмы программирования и их реализация в языке Python. Помимо объектно-ориентированного программирования были применены элементы функционального программирования. Был разработан код, в котором были описаны пять классов – *Edition*, наследники которого – *Book* и *Newspaper*; *BookList* и *NewspaperList*, родителем которых выступил стандартный класс List. Были написаны методы для работы с экземплярами этих классов, причём важной их объединяющей особенностью служил контроль типов с помощью функции *isinstance*, с вызовом ошибки с помощью *raise* при необходимости. Так же были учтены особенности наследования и при переопределении метода родительского класса использовалась функция *super*() для избежания дублирования кода.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class Edition:
    def __init__(self, name, price, age_limit, style):
        if isinstance(name, str) and isinstance(price, int) and price
> 0 and isinstance(age_limit,

int) and age_limit > 0 and style in (
        'c', 'b'):
            self.name = name
            self.price = price
            self.age_limit = age_limit
            self.style = style
        else:
            raise ValueError('Invalid value')


class Book(Edition):
    def __init__(self, name, price, age_limit, style, author,
hardcover, pages):
        super().__init__(name, price, age_limit, style)
        if isinstance(author, str) and isinstance(hardcover, bool)
and isinstance(pages, int) and pages > 0:
            self.author = author
            self.hardcover = hardcover
            self.pages = pages
        else:
            raise ValueError('Invalid value')

    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 isinstance(online_edition, bool) and isinstance(country,
str) and isinstance(frequency,

int) and frequency > 0:
            self.online_edition = online_edition
            self.country = country
            self.frequency = frequency
        else:
            raise ValueError('Invalid value')

    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, __object):
        if isinstance(__object, Book):
            super().append(__object)
        else:
```

```
            raise TypeError(f'Invalid type {type(__object)}')

    def total_pages(self):
        return sum(map(lambda b: b.pages, self))

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


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

    def extend(self, iterable):
        super().extend(filter(lambda x: isinstance(x, Newspaper),
iterable))

    def print_age(self):
        print(min(map(lambda newspaper: newspaper.age_limit, self)))

    def print_total_price(self):
        print(sum(map(lambda newspaper: newspaper.price, self)))
```