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. 3342 _____ Пушко К.Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Изучение парадигм программирования. Написать программу с использованием концепции ООП.

Задание

Вариант 4.

Базовый класс — печатное издание *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'.

Класс книга – *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.

Класс газета – *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*. Конструктор принимает *name*, *price*, *age_limit*, *style* в качестве параметров, название, цена, возрастное ограничение стиль соответственно, параметры присваиваются полям класса. Производится проверка на тип, у всех параметров, а также на значение: цена положительное число, стиль: либо *c* либо *b*. В случае несоответствия предъявленным требованиям вызывается исключение ValueError с сообщением: “Invalid value”.

Класс *Book* наследуется от класса *Edition*. Конструктор принимает name, *price, age_limit, style, author, hardcover, pages* в качестве параметров, название, цена, возрастное ограничение, стиль, автор, твердый переплет, количество страниц соответственно. Поля *name, price, age_limit, style* передаются конструктору родительского класса. Проводится проверка на соответствие типам оставшихся параметров, а также на положительность числа страниц. В случае несоответствия предъявленным требованиям вызывается исключение ValueError с сообщением: “Invalid value”. Затем параметры *author, hardcover, pages* присваиваются полям класса. Переопределяется метод *__str__* для приведению класса к типу string, например при помещении экземпляра класса в функцию print(), переопределяется метод *__eq__*, в котором сравниваются авторы и страны издания объектов, это необходимо для понимания являются ли данные экземпляры одними объектами по смыслу.

Класс *Newspaper* наследуется от класса *Edition*. Конструктор принимает *name, price, age_limit, style, online_edition, country, frequency* в качестве параметров, название, цена, возрастное ограничение, стиль, онлайн ли издание, страна, частота выпуска соответственно. Поля *name, price, age_limit, style* передаются конструктору родительского класса. Проводится проверка на соответствие типам оставшихся параметров, а также на положительность частоты издания. В случае несоответствия предъявленным требованиям вызывается исключение ValueError с сообщением: “Invalid value”. Затем параметры *online_edition, country,* frequency присваиваются полям классов. Переопределяется метод *__str__* для приведению класса к типу string, например

при помещении экземпляра класса в функцию print(), переопределяется метод *__eq__*, в котором сравниваются авторы и страны издания объектов, это необходимо для понимания являются ли данные экземпляры одними объектами по смыслу.

Класс *BookList* наследуется от класса *list*. В конструктор передается имя списка, в нем вызывается родительский конструктор, а затем присваивается параметр *name*. Переопределяется метод *append*, в котором проверяется тип добавляемого объекта, в случае несоответствия, вызывается TypeError, иначе вызывается *append* у родительского метода. Метод *total_pages* возвращает сумму всех страниц. Метод *print_count* печатает количество книг в списке.

Класс *NewspaperList* наследуется от класса *list*. В конструктор передается имя списка, в нем вызывается родительский конструктор, а затем присваивается параметр *name*. Переопределяется метод *extend*, в цикле проверяется все ли элементы *iterable* корректного типа, в случае несоответствия метод завершается, иначе вызывается родительский *extend*. Метод *print_age* печатает минимальное возрастное ограничение из всех элементов списка. Метод *print_total_price* печатает суммарную стоимость газет из списка.

Разработанный программный код см. в приложении А.

Выводы

Были изучены парадигмы программирования. Написана программа с использованием концепции ООП.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class Edition:

    def isNatural(self,x):
        return isinstance(x,int) and x>0

    def __init__(self, name, price, age_limit, style):
        if isinstance(name,str) and self.isNatural(price) and
self.isNatural(age_limit) 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
self.isNatural(pages):
            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}, ст
```

```
и л ь {self.style}, а в т о р {self.author}, т в е р д ы й  п е р е п л е т {self.hardcover}, к о л и ч е с т в о  с т р а н и ц {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 self.isNatural(frequency):
                self.online_edition = online_edition
                self.country = country
                self.frequency = frequency
            else:
                raise ValueError('Invalid value')

        def __str__(self):
            return f'Newspaper: н а з в а н и е {self.name}, ц е н а {self.price}, в о з р а с т н о е  о г р а н и ч е н и е {self.age_limit}, с т и л ь {self.style}, и н т е р н е т  и з д а н и е {self.online_edition}, с т р а н а {self.country}, п е р и о д и ч н о с т ь {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):
            return sum(i.pages for i in 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([i  for  i  in  iterable  if  isinstance(i,
Newspaper)])

        def print_age(self):
            print(min(i.age_limit for i in self))

        def print_total_price(self):
            print(sum(i.price for i in self))
```