**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}, 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 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: 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):

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))