**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. 3344 |  | Бубякина Ю.В. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Получить представление о работе ООП в языке Python.

## Задание.

**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:

1. Вызвать конструктор базового класса.
2. 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:

1. Вызвать конструктор базового класса.
2. 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



Рисунок 1 – Изображение иерархии классов

1. Методы классов, унаследованных у Edition:

\_\_init\_\_() – Принимаются параметры и происходит их проверка на корректность

\_\_str\_\_() – Преобразование данные в строку и возвращает её

\_\_eq\_\_() – Сравнивает два объекта данного типа

1. Методы классов, унаследованных у list:

\_\_init\_\_() – Принимает параметр и проверяет его на корректность

append() – Добавляет в список элемент данного типа

total\_pages() – Возвращает количество всех страниц из всех книг

print\_count() – Возвращает количество книг

extend() – Расширяет список другим списком, если в нём хранятся значения данного типа

print\_age() – Выводит минимальное возрастное ограничение среди газет

print\_total\_price() – Выводит сумму всех газет

1. Метод \_\_str\_\_() будет применяться при вызове str(<экземпляр класса>)

Метод \_\_eq\_\_() будет применяться при сравнении экземпляров (==)

1. Переопределённые методы будут работать, так как у них сохраняется вся функциональность, к которой просто добавляется проверка на входные данные.

## Выводы

Получен навык работы с ООП. Изучены особенности реализации данной методологии программирования в языке Python.

# Приложение А Исходный код программы

Название файла: Bubyakina\_Yuliya\_lb1.py

class Edition:

def \_\_init\_\_(self, name, price, age\_limit, style):

if not (isinstance(name, str) and isinstance(price, int) and price > 0 and

isinstance(age\_limit, int) and age\_limit > 0 and (style == 'c' or style == 'b')):

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) and isinstance(hardcover, bool) and isinstance(pages, int) and 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}, '

f'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) and isinstance(country, str) and isinstance(frequency, int) and 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},'

f' стиль {self.style}, интернет издание {self.online\_edition}, страна {self.country}, периодичность {self.frequency}.')

def \_\_eq\_\_(self, other):

return self.country == other.country and self.name == other.name

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 <тип объекта {type(\_\_object)}>')

def total\_pages(self):

return sum(x.pages for x 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(x for x in \_\_iterable if isinstance(x, Newspaper))

def print\_age(self):

print(min(x.age\_limit for x in self))

def print\_total\_price(self):

print(sum(x.price for x in self))