**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 |  | Легалов В. В. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Изучить основы программирования, включая работу с классами, исключениями и их применение в Python. С использованием этих знаний реализовать программу для создания экземпляров классов.

## 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:

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:

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

Конструктор класса Edition принимает параметры name, price, age\_limit и style для названия, цены, возрастного ограничения и стиля соответственно. Значения параметров присваиваются соответствующим полям класса. Все параметры проходят проверку на тип данных, а также на допустимые значения. В случае несоответствия заданным требованиям, вызывается исключение ValueError с сообщением: “Invalid value”.

От класса Edition наследуется касс Book. Конструктор класса Book принимает параметры name, price, age\_limit, style, author, hardcover и pages для названия, цены, возрастного ограничения, стиля, автора, типа обложки (мягкий/твердый) и количества страниц соответственно. Поля name, price, age\_limit, style передаются конструктору родительского класса. При инициализации происходит проверка типов остальных параметров. Метод \_\_str\_\_() переопределен для конвертации класса в строку, что позволяет корректно выводить экземпляр класса с помощью функции print() или преобразования str(). Также переопределен метод \_\_eq\_\_(), в котором происходит сравнение авторов и стран издания.

От класса Edition наследуется класс Newspaper. Конструктор класса принимает параметры: name, price, age\_limit, style, online\_edition, country, frequency, которые соответствуют названию, цене, возрастному ограничению, стилю, наличию онлайн-издания, стране и частоте выпуска соответственно. Поля name, price, age\_limit и style передаются конструктору родительского класса. При инициализации происходит проверка типов остальных параметров. Метод \_\_str\_\_() переопределен для конвертации класса в строку. Также переопределяется метод eq, в котором происходит сравнение авторов и стран издания объектов.

Класс BookList наследуется от класса list. В его конструкторе сначала вызывается конструктор родительского класса, а затем присваивается параметр name. Метод append переопределён так, что он проверяет тип добавляемого объекта: если тип не соответствует ожидаемому, возникает ошибка TypeError, в противном случае вызывается метод append у родительского класса. Метод total\_pages возвращает общее количество страниц всех книг в списке. Метод print\_count выводит на экран количество книг в списке.

Класс NewspaperList наследуется от класса list. При создании экземпляра передается имя списка, которое затем присваивается переменной name. Переопределен метод extend, который добавляет только элементы определенного типа. Метод print\_age выводит минимальное требуемое возрастное ограничение для всех элементов списка. Метод print\_total\_price выводит общую стоимость всех газет в списке.

Разработанный программный код см. в приложении А.

## Выводы

Были исследованы парадигмы программирования. Была разработана программа, в которой применяются основные принципы и концепции ООП.

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 isinstance(style, str) 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, p\_object):

try:

if isinstance(p\_object, Book):

super().append(p\_object)

else:

raise TypeError

except TypeError:

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))