**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**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3342 |  | Смирнова Е.С. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## **Цель работы**

Научиться использовать классы в Python, а также реализовать поставленную задачу используя принципы объектно-ориентированного программирования.

## **Задание**

(Вариант 4)

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:

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**

В ходе лабораторной работы были реализованы 5 классов.

Класс Edition, в котором определен конструктор, а также с помощью конструкции try-except осуществляется проверка данных, поданных в этот конструктор.

От класса Edition наследуются классы Book и Newspaper. В каждом из соответствующих классов также реализован конструктор, в котором используется функция super() для определения полей родительского класса. Также в этих классах переопределены методы \_\_str\_\_() и \_\_eq\_\_(). Первый метод отвечает за поведение функции print(), когда в качестве аргумента ей передается экземпляр одного из этих классов. Метод \_\_eq\_\_() переопределяет работу оператора сравнения. При сравнении двух объектов одного из этих классов выполняется проверка на равенство только заранее указанных полей. Если поля совпадают, результатом сравнения будет True, иначе False.

Следующие 2 класса, а именно BookList и NewspaperList наследуются от встроенного типа данных list. В них также есть конструкторы для указания имени конкретного списка, а также реализованы некоторые методы. Для класса BookList переопределен метод append(), с помощью try-except и фукнции isinstance() осуществляется проверка аргумента передаваемого методу append() на принадлежность классу Book. Если аргумента является экземпляром соответствующего класса, то с помощью функции super() вызывается метод append() родительского класса, иначе вызывается исключение TypeError.

Также для класса BookList определены методы total\_pages(), который выводит суммарное количество страниц всех книг, содержащемся в данном списке, и print\_count(), который печатает количество элементов в списке. В классе NewspaperList переопределен метод extend. С помощью функций all() и isinstance() определяется являются ли все элементы добавляемого списка экземплярами класса Newspaper. Если функция all() вернула True, то с помощью функции super() вызывается метод extend класса list, иначе ничего не происходит. Также реализован метод print\_age(), который с помощью функции min и списочного выражения определяет и выводит самое низкое возрастное ограничение, и метод print\_total\_price(), который печатает сумму цен всех книг.

Разработанный программный код см. в приложении А.

## **Выводы**

В ходе лабораторной работы были освоены некоторые парадигмы программирования, а также получены практические навыки по работе с классами.

# **ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: main.py

class Edition:

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

try:

if 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'):

self.name = name

self.price = price

self.age\_limit = age\_limit

self.style = style

else:

raise ValueError

except ValueError:

raise ValueError('Invalid value')

class Book(Edition):

def \_\_init\_\_(self, name: str, price: int, age\_limit: int, style: str, author: str, hardcover: bool, pages: int):

super().\_\_init\_\_(name, price, age\_limit, style)

try:

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

except ValueError:

raise ValueError('Invalid value')

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},' \

f' количество страниц {self.pages}.'

def \_\_repr\_\_(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},' \

f' 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: str, price: int, age\_limit: int, style: str, online\_edition: bool, country: str, frequency: int, ):

super().\_\_init\_\_(name, price, age\_limit, style)

try:

if isinstance(country, str) and isinstance(online\_edition, bool) and (isinstance(frequency, int) and frequency > 0):

self.online\_edition = online\_edition

self.country = country

self.frequency = frequency

else:

raise ValueError

except ValueError:

raise ValueError('Invalid value')

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},' \

f' периодичность {self.frequency}.'

def \_\_repr\_\_(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},' \

f' 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: Book):

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))