**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3343 |  | Гельман П.Е. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Цель данной лабораторной работы — изучить основы объектно-ориентированного программирования на примере Python. Основное внимание уделено работе с классами, созданию методов и функций для классов, пониманию принципов наследования, переопределения методов и работы с методом super().

## 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:

* 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 лабораторной работе необходимо создать классы с определёнными методами, которые представляют собой фигуры с определёнными параметрами и списки для хранения этих фигур.

Класс Edition является родительским для классов Book и Newspaper и хранит в себе информацию о названии, цене, возрастном ограничении и стиле объекта. При создании экземпляра класса проверяется, удовлетворяют ли переданные в конструктор параметры требованиям, иначе выводится исключение ValueError.

Класс Book описывает книгу. Поля этого класса: автор книги, информация о жесткости переплета, количество страниц. Также реализованы метод, который выводит информацию об объекте и метод, который сравнивает два объекта этого класса по названию и автору.

Класс Newspaper описывает газету. Он содержит информацию об интернет издании газеты, стране, периодичности выпуска. Добавлены методы для вывода информации о газете и сравнения двух объектов класса, если равны название и страна — True, иначе — False.

Класс BookList — список книг, наследуется от класса list. В классе переопределяется метод append() списка: если объект — книга, элемент добавляется в список, иначе исключение TypeError. Метод total\_pages() возвращает сумму всех страниц всех имеющихся книг. Метод print\_count() выводит количество книг.

Класс NewspaperList — список газет — наследуется от класса list. Переопределен метод extend() списка: если элемент iterable — объект класса Newspaper, этот элемент добавляется в список. Метод print\_age() выводит самое низкое возрастное ограничение среди всех газет. Метод print\_total\_price() считает и выводит общую цену всех газет.

Метод \_\_str\_\_ является специальным методом, предназначенным для представления строкового представления объекта. Когда вызывается функция str() или встроенная функция print() для объекта, Python автоматически вызывает метод \_\_str\_\_, если он определен, чтобы получить строковое представление объекта.

Метод \_\_eq\_\_ в Python используется для определения логики сравнения двух объектов на равенство. Когда переопределяется метод \_\_eq\_\_ в классе, нужно определить, как объекты этого класса будут сравниваться при использовании оператора "==". Внутри метода \_\_eq\_\_ можно указать любую логику сравнения, которая необходима для структурного или значимого сравнения двух экземпляров класса.

При вызове выражения obj1 == obj2, Python автоматически вызывает метод \_\_eq\_\_ для объекта obj1 с передачей второго объекта obj2 в качестве аргумента. Метод \_\_eq\_\_ должен вернуть True, если объекты равны, и False, если они не равны.

Переопределенные методы append и extend в классе работают с помощью вызова super(). Это позволяет обращаться к методам append и extend из класса list и корректно добавлять объекты. Это сделано для того, чтобы гарантировать правильное поведение этих методов.

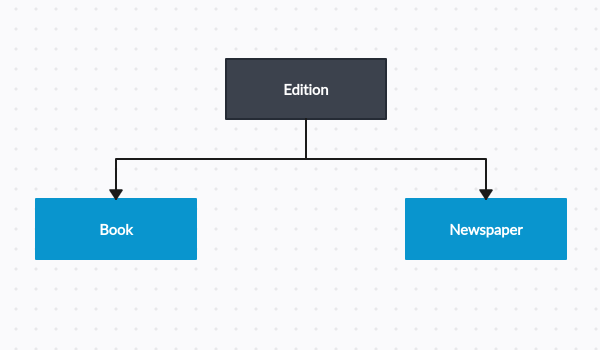


Рисунок 1 – Иерархия классов фигур

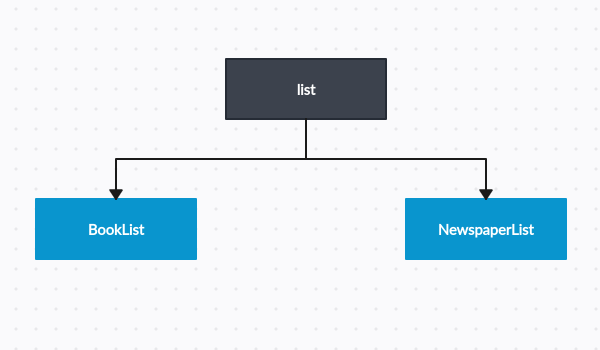


Рисунок 2 – Иерархия классов списков фигур

## Выводы

В ходе выполнения данной лабораторной работы были изучены основы объектно-ориентированного программирования на примере языка Python. Основной упор был сделан на работу с классами, создание методов и функций для классов, понимание принципов наследования, переопределения методов и использования метода super().

# Приложение А Исходный код программы

Название файла: main.py

class Edition:

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

if not isinstance(name, str) or not isinstance(price, int) or \

not isinstance(age\_limit, int) or not isinstance(style, str):

raise ValueError('Invalid value')

if price <= 0 or age\_limit <= 0 or style not in ['c', '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) or not isinstance(hardcover, bool) or not isinstance(pages, int):

raise ValueError('Invalid value')

if 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}, 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) or not isinstance(country, str) or not isinstance(frequency, int):

raise ValueError('Invalid value')

if 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}, 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):

total = 0

for book in self:

total += book.pages

return total

def print\_count(self):

print(len(self))

class NewspaperList(list):

def \_\_init\_\_(self, name):

super().\_\_init\_\_()

self.name = name

def extend(self,iterable):

for i in iterable:

if isinstance(i, Newspaper):

super().append(i)

def print\_age(self):

ages = [newspaper.age\_limit for newspaper in self]

if ages:

min\_age = min(ages)

print(min\_age)

def print\_total\_price(self):

cost = 0

for newspaper in self:

cost += newspaper.price

print(cost)