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

Студент гр. 3341 _____ Трофимов В.О.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Целью работы является написание программы на Python, содержащая классы и их иерархию, методы для работы с этими классами, а также классы, наследуемые от стандартных классов языка Python. В ходе работы следует изучить основные 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.

Основные теоретические положения

1. Классы и объекты: в Python все является объектом, даже базовые типы данных. Классы определяются с помощью ключевого слова class, а объекты создаются путем вызова конструктора класса.

2. Атрибуты и методы: классы могут содержать атрибуты (переменные) и методы (функции), которые определяют поведение объектов.

3. Наследование: классы могут наследовать свойства и методы других классов. Наследование позволяет создавать иерархии классов и повторно использовать код.

4. Полиморфизм: классы могут переопределять методы родительских классов для изменения их поведения. Полиморфизм позволяет одному методу иметь различное поведение в разных классах.

5. Инкапсуляция: классы могут объединять данные (атрибуты) и методы внутри себя и скрывать их от внешнего мира. Инкапсуляция позволяет обеспечить безопасность и четкость кода.

Выполнение работы

1. Создание класса Edition, который содержит поля name, price, age_limit, style. В конструкторе происходит проверка параметров на условия, если проверку не проходят генерируется исключение ValueError(“Invalid value”).

2. Создание класса Book, наследуемого от Edition. В конструкторе появляются новые поля author, hardcover, pages, которые проверяются на условия, если проверку не проходят генерируется исключение ValueError(“Invalid value”). В классе переопределены методы основного класса object: __str__ и __eq__.

3. Создание класса Newspaper, наследуемого от Edition. В конструкторе появляются новые поля online_edition, country, frequency, которые проверяются на условия, если проверку не проходят генерируется исключение ValueError(“Invalid value”). В классе переопределены методы основного класса object: __str__ и __eq__.

4. Создание класса Booklist, наследуемого от list. У класса есть атрибут name, который инициализируется в методе __init__. В классе переопределён метод append, теперь он добавляет только объекты типа Book в список. Если объект не является экземпляром класса Book, генерируется исключение TypeError. Метод total_pages возвращает сумму страниц всех книг в списке. Он использует генераторное выражение для перебора всех элементов и вычисления суммы страниц. Метод print_count выводит количество книг в списке с помощью функции len(self), которая возвращает количество элементов в списке.

5. Класс NewspaperList также наследуется от встроенного типа данных list. У класса есть атрибут name, который инициализируется в методе __init__. В классе определён метод extend, который позволяет добавлять к списку газет другие газеты из переданного итерируемого объекта. Он проверяет, что все элементы из переданного итерируемого объекта являются экземплярами класса Newspaper, и только в этом случае расширяет список. Метод print_age выводит минимальный возрастной лимит из всех газет в списке. Он использует генераторное выражение для перебора всех элементов и нахождения

минимального возрастного лимита. Метод print_total_price выводит общую цену всех газет в списке. Он вычисляет сумму цен всех газет в списке с помощью генераторного выражения.

1. Иерархия классов:

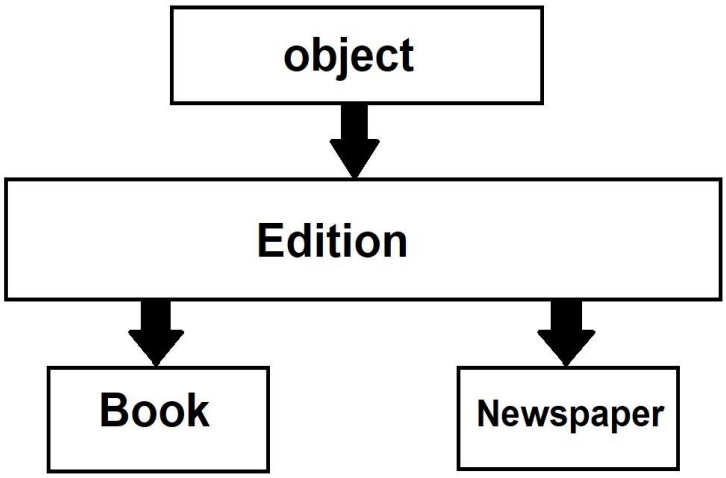

Схема 1

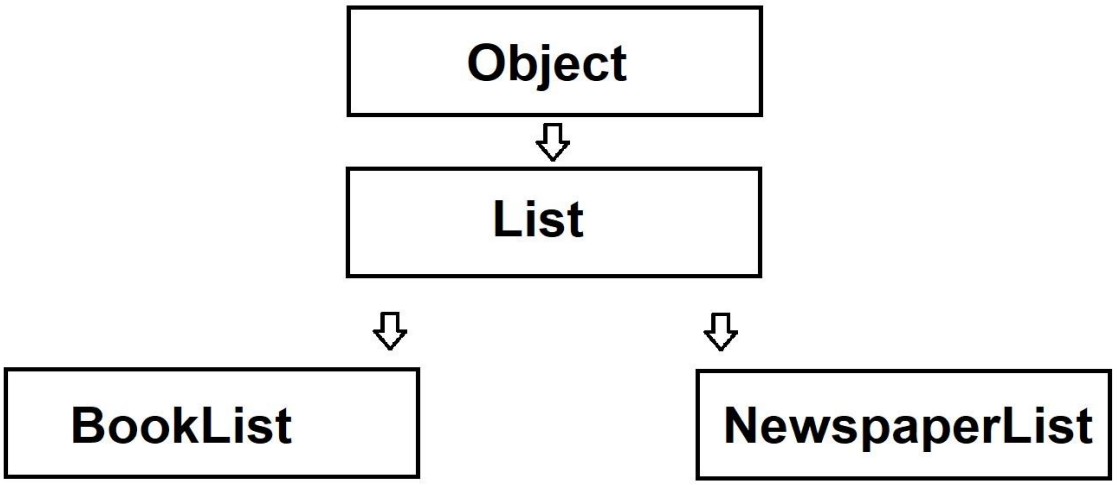

Схема 2

2. Методы, которые были переопределены:

1. __init__ - во всех классах для определения полей класса.

2. __str__ - в классах Book, Newspaper для определённого представления о классе.

3. __eq__ - в классах для сравнения их полей в Book: name, author; в Newspaper: name, counrty.

4. append –в классах для того, чтобы в классы были добавлены только их представители.

5. extend –в классах, чтобы в классы были добавлены только их представители

3. Случаи использования методов __str__() и __eq__():

Метод str() будет использован для получения строкового представления объекта, например при вызове функции print().

Метод eq() будет использован для сравнения двух объектов.

4. Переопределенные методы класса list для BookList и NewspaperList будут работать, так как они написаны без использования методов класса list, и не зависят от него, пример:

Пример для BookList:

```
book1 = Book("Book1", 20, 18, "c", "Author1", True, 300)
book2 = Book("Book2", 15, 16, "b", "Author2", False, 200)
book_list = BookList("My Books")
book_list.append(book1)
book_list.append(book2)
print(book_list.total_pages()) #500
print(len(book_list #2
```

Пример для NewspaperList:

```
newspaper1 = Newspaper("Newspaper1", 5, 16, "b", True, "USA", 7)
newspaper2 = Newspaper("Newspaper2", 3, 14, "c", False, "UK", 5)
newspaper_list = NewspaperList("My Newspapers")
newspaper_list.extend([newspaper1, newspaper2])
print(len(newspaper_list)) #2
```

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	book1 = Book("Check1", 20, 18, "b", "F. Scott Fitzgerald", True, 256) book2 = Book("Check2", 25, 10, "c", "J.K. Rowling", True, 332) newspaper1 = Newspaper("Check3", 5, 16, "b", True, "USA", 7) newspaper2 = Newspaper("Check4", 3, 15, "c", True, "UK", 5) booklist = BookList("My Book List") newspaperlist = NewspaperList("My Newspaper List") booklist.append(book1) booklist.append(book2) newspaperlist.extend([newspaper1, newspaper2]) print(booklist.total_pages()) booklist.print_count() newspaperlist.print_age() newspaperlist.print_total_price()	588 2 15 8
2.	try: book1 = Book("Check1", 20, 0, "b", "F. Scott Fitzgerald", 2, 256) book2 = Book("Check2", 25, 0, "c", "J.K. Rowling", 2, 332) newspaper1 = Newspaper("Check3", 5, 0, "b", 2,	Traceback (most recent call last): File "d:\Desktop\Projects\Python\Trofimov_Vladislav_lb1\src\main.py", line 98, in <module> raise ValueError("Invalid

	"USA", 7) newspaper2 = Newspaper("Check4", 3, 0, "c", 2, "UK", 5) booklist = BookList("My Book List") newspaperlist = NewspaperList("My Newspaper List") booklist.append(book1) booklist.append(book2) newspaperlist.extend([newspaper1, newspaper2]) print(booklist.total_pages()) booklist.print_count() newspaperlist.print_age() newspaperlist.print_total_price() except ValueError: raise ValueError("Invalid value")	value") ValueError: Invalid value

Выводы

Цель работы была достигнута, написана программа на Python и изучены основные принципы объектно-ориентированного программирования. В программе были созданы классы Edition, Book, Newspaper, BookList и NewspaperList, которые демонстрируют иерархию классов и принципы объектно-ориентированного программирования. Класс Edition является базовым классом для классов Book и Newspaper, и содержит основные атрибуты (название, цена, возрастное ограничение, стиль), которые наследуются и дополняются в дочерних классах. Классы Book и Newspaper расширяют класс Edition, добавляя свои собственные атрибуты (автор, твердый переплет, количество страниц для Book; интернет издание, страна, периодичность для Newspaper) и методы для их работы (str для вывода информации о книге или газете, eq для сравнения книг или газет). Классы BookList и NewspaperList наследуются от стандартного класса list и предоставляют специфичные методы для работы со списками книг и газет, такие как добавление книги/газеты в список, подсчет общего количества страниц у книг, вывод количества книг в списке, вывод минимального возрастного ограничения и общей цены газет в списке.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class Edition:
    def __init__(self,name,price,age_limit,style):
        if not all([isinstance(name,str), isinstance(price,int),price > 0, isinstance(age_limit, int),age_limit > 0,style 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 all([isinstance(author,str), isinstance(hardcover,bool),isinstance(pages,int),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):
        if all([self.name == other.name,self.author == other.author]):
            return True
        return False

class Newspaper(Edition):
    def __init__(self, name, price, age_limit, style, online_edition, country, frequency):
        super().__init__(name, price, age_limit, style)
        if not all([isinstance(online_edition, bool), isinstance(country,str), isinstance(frequency, int), 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):
            if all([self.name == other.name, self.country ==
other.country]):
                return True
            return False

    class BookList(list):
        def __init__(self, name):
            super().__init__(self)
            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(item.pages for item in self)

        def print_count(self):
            print(len(self))

    class NewspaperList(list):
        def __init__(self, name):
            super().__init__(self)
            self.name = name

        def extend(self, iterable):
            if all(isinstance(item,Newspaper) for item in iterable):
                super().extend(iterable)

        def print_age(self):
            min_age_limit = min(newspaper.age_limit for newspaper in
self)
            print(min_age_limit)

        def print_total_price(self):
            all_price = sum(newspaper.price for newspaper in self)
            print(all_price)
```