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. 3341 _____ Яковлева А.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Целью данной работы является:

- изучение парадигм программирования
- создание иерархии классов для представления книг, газет и их списков
- обработка исключительных ситуаций
- переопределение методов списка

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

Создаем класс *Edition*, содержащий поля *name, price, age_limit, style*. При инициализации объекта функцией *isinstance* проверяем соответствие значений заданному типу (*str* или *int*), а также проверяем значение *price* и *style,* при несоответствии выбрасываем исключение *ValueError* с текстом '*Invalid value*'.

Создаем класс *Book*, который наследуется от класса *Edition* и добавляем поля *author, hardcover, pages*. При инициализации объекта вызываем конструктор родительского класса, затем функцией *isinstance* проверяем соответствие значений заданному типу (*str, bool* или *int*), а также проверяем значение *pages,* при несоответствии выбрасываем исключение *ValueError* с текстом '*Invalid value*'. Переопределяем методы *str* и *eq*. Метод *str* возвращает строку в определённом виде, *eq* сравнивает два объекта класса *Book*.

Создаем класс *Newspaper*, который также наследуется от класса *Edition* и добавляет поля *online_edition, country, frequency*. При инициализации объекта вызываем конструктор родительского класса, затем функцией *isinstance* проверяем соответствие значений заданному типу (*str, bool* или *int*), а также проверяем значение *frequency,* при несоответствии выбрасываем исключение *ValueError* с текстом '*Invalid value*'. Переопределяем методы *str* и *eq*. Метод *str* возвращает строку в определённом виде, *eq* сравнивает два объекта класса *Newspaper.*

Создаем класс *BookList*, который наследуется от списка, переопределяем методы *init*, *append*, добавляем методы *total_pages, print_count*. В методе *append* при соответствии *p_object* классу *Book* добавляем *p_object* в список функцией родительского класса *super().append(p_object),* иначе выбрасывается исключение *TypeError* с текстом: *Invalid type <тип_объекта p_object>*. Метод *total_pages* возвращает сумму всех страниц всех книг, посчитанную с помощью функции *sum*, метод *print_count* выводит количество книг, т.е. количество элементов в списке.

Создаем класс *NewspaperList*, аналогично *BookList*, переопределяем методы *init, extend*, добавляем методы *print_age, print_total_price*. В методе *extend* при помощи *filter* и лямбда-функции проверяется соответствии элементов классу *Newspaper,* полученный список добавляется к текущему функцией родительского

класса *super().extend.* Метод *print_age* выводит наименьшее значение возрастного ограничения из всех газет, метод *print_total_price* выводит сумму стоимости всех газет.

1. Изображение иерархии классов:

Edition List

/ \ / \

Book Newspaper BookList NewspaperList

2. Переопределённые методы (в том числе методы класса object):

- метод *__init__*(): переопределен в каждом классе для инициализации полей.
- метод *__str__*(): переопределен для возвращения строкового представления объекта.
- метод *__eq__*(): возвращает True, если два объекта класса равны и False иначе.
- метод *append(p_object)*: переопределение метода *append(p_object)* списка.
- метод *extend(iterable)*: переопределение метода *extend(iterable)* списка.

3. Метод `*__str__*()` будет использован, когда объект класса вызывается как аргумент функции `str()`, чтобы получить его строковое представление в определённом виде. Метод `*__eq__()*` используется для сравнения двух объектов класса.

4. Переопределенные методы класса *list* для *BookList* и *NespaperList* будут работать, т.к. *BookList* и *NespaperList* наследники *list.*

Пример:

```
books = BookList('books')
book1 = Book('Name1', 90, 12, 'b', 'Ivanov', True, 250)
books.append(book1) # добавление книги
newspapers = NewspaperList('newspapers')
newspaper1 = Newspaper('Name1', 190, 12, 'b', True, 'Russia', 7)
newspaper2 = Newspaper('Name2', 90, 15, 'b', True, 'England', 7)
newspapers.extend([newspaper1, '123', newspaper2]) # добавятся элементы newspaper1 и newspaper2, не добавится '123', т.к. он не является объектом класса Newspaper
```

```

Разработанный программный код см. в приложении А.
```

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	book1 = Book('Name', -90, 12, 'b', 'Ivanov', True, 250)	ValueError: Invalid value	Проверка обработки ошибок
2.	book1 = Book('Name', 90, 12, 'b', 'Ivanov', True, 250)book2 = Book('Name', 90, 12, 'b', 'Ivanov', True, 250) book3 = Book('Different name', 90, 12, 'b', 'Ivanov', True, 250) print(book1.__str__()) print(book1.__eq__(book2)) print(book3.__eq__(book2))	Book: название Name, цена 90, возрастное ограничение 12, стиль b, автор Ivanov, твердый переплет True, количество страниц 250. True False	Проверка методов класса Book
3.	newspaper1 = Newspaper('Name', 90, 12, 'b', True, 'Russia', 7) newspaper2 = Newspaper('Name', 90, 12, 'b', True, 'Russia', 7) newspaper3 = Newspaper('Name', 90, 12, 'b', True, 'England', 7) print(newspaper1.__str__()) print(newspaper1.__eq__(newspaper2)) print(newspaper1.__eq__(newspaper3))	Newspaper: название Name, цена 90, возрастное ограничение 12, стиль b, интернет издание True, страна Russia, периодичность 7. True False	Проверка методов класса Newspaper
4.	books = BookList('books') book1 = Book('Name1', 90, 12, 'b', 'Ivanov', True, 250) book2 = Book('Name2', 90, 12, 'b', 'Ivanov', True, 130)	380 2	Проверка методов класса BookList

	books.append(book1) books.append(book2) print(books.total_pages()) books.print_count()		
5.	newspapers = NewspaperList('newspapers') newspaper1 = Newspaper('Name1', 190, 12, 'b', True, 'Russia', 7) newspaper2 = Newspaper('Name2', 90, 15, 'b', True, 'England', 7) newspapers.extend([newspaper1,'123', newspaper2]) newspapers.print_age() newspapers.print_total_price()	12 280	Проверка методов класса NewspaperList

Выводы

Были изучены парадигмы программирования, с помощью наследования создана иерархия классов для представления книг, газет и их списков, обработаны исключительные ситуации, переопределены методы списка.

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 (style == 'c' or style == '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.author == other.author and self.name == other.name


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):
            if isinstance(p_object, Book):
                super().append(p_object)
            else:
                raise TypeError(f"Invalid type", type(p_object))

        def total_pages(self):
            return sum([book.pages for book 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(list(filter(lambda element: isinstance(element, Newspaper), iterable)))

        def print_age(self):
            print(min([newspaper.age_limit for newspaper in self]))

        def print_total_price(self):
            print(sum([newspaper.price for newspaper in self]))
```