**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. 3341 |  | Моисеева А.Е. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Целью является анализ концепций парадигм программирования и их практическое применение.

Основные задачи:

1.Изучить парадигмы программирования.

2.Детально изучить реализацию функционального программирования на языке Python, научиться применять полученные знания на практике.

3.Разработать программу, которая использует классы и модифицированные методы базового класса для реализации задачи с определённым условием для работы над данными.

## 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.

## Основные теоретические положения

Объектно-ориентированное программирование — это парадигма программирования, которая использует "объекты" - структуры данных, состоящие из полей данных и методов вместе с их взаимодействиями, для написания программ. Python поддерживает множество парадигм программирования, включая процедурную, функциональную и объектно-ориентированную.

Основными составляющими в программах с объектно-ориентированной парадигмой являются классы и объекты. Класс – это своего рода шаблон, из которого могут быть созданы индивидуальные экземпляры, называемые объектами. Эти объекты содержат как состояния (обычно представленные переменными, называемыми атрибутами), так и поведения (функции, связанные с этим объектом, называемые методами).

Инкапсуляция означает ограничение доступа к определенным компонентам объекта и предотвращение внешнего вмешательства или нежелательного использования. В Python, инкапсуляция реализуется с помощью защищенных атрибутов и методов, которые начинаются с одного или двух символов подчеркивания.

Наследование позволяет новому классу перенимать атрибуты и методы существующего класса. В Python, наследование осуществляется путем передачи родительского класса в качестве аргумента при определении нового класса.

Полиморфизм — это способность использовать общий интерфейс для разных базовых форм (данных или методов). В Python это достигается благодаря тому, что объекты разных классов могут быть обработаны одним и тем же способом, если эти классы реализуют определенные.

Объектно-ориентированный подход к программированию облегчает работы с большими и сложными системами, программы строятся из отдельных частей, которые взаимодействуют друг с другом. ООП также позволяет разработчикам сосредоточиться на высокоуровневых операциях, избегая деталей реализации.

## Выполнение работы

1.Создаётся класс Edition, который содержит атрибуты name, price, age\_limit, style. При создании объекта класса производится проверка входных данных (name должен быть строкой, price и age\_limit - целым положительным числом, style - строкой вида “c” или “b”).

2.Создаётся класс Book, который наследуется от Edition и добавляет к прошлым атрибуты author, hardcover, pages. При создании объекта класса вызываем \_\_init\_\_ родительского класса, и проводится проверка данных (author должен быть строкой, hardcover принимать значение True или False, pages – целым положительным числом). Переопределяются методы \_\_str\_\_ и \_\_eq\_\_. Метод \_\_str\_\_ возвращает строку в некотором заданном виде. Метод \_\_eq\_\_ сравнивает два значения класса Book (равны, если совпадают значения атрибутов name и author).

3. Создаётся класс Newspaper, который наследуется от Edition и добавляет к прошлым атрибуты online\_edition, country, frequency. При создании объекта вызываем \_\_init\_\_ родительского класса, затем проводится проверка (online\_edition должен принимать значение True или False country – быть строкой, frequency – целым положительным числом). Переопределяются методы \_\_str\_\_ и \_\_eq\_\_. Метод \_\_str\_\_ возвращает строку в некотором заданном виде. Метод \_\_eq\_\_ сравнивает два значения класса Newspaper (равны, если совпадают значения атрибутов name и country).

4.Создаётся класс BookList, который наследуется от класса list и добавляет методы \_\_init\_\_, append, total\_pages, print\_count. Метод append позволяет добавлять в список только объекты класса Book, метод total\_pages возвращает сумму страниц всех книг, метод print\_count выводит общее количество книг.

5. Создаётся класс NewspaperList, который наследуется от класса list и добавляет методы \_\_init\_\_, extend, print\_age, print\_total\_price. Метод extend позволяет добавлять в список только объекты класса Newspaper, метод print\_age выводит наименьшее значение возрастного ограничения из всех газет, метод print\_total\_price выводит сумму стоимости всех газет.

Этот код создаёт структуру классов, представляющих издания, а также списки для их хранения по категориям. Для каждого типа издания определены специфические атрибуты и функционал.

1. Изображение иерархии классов:

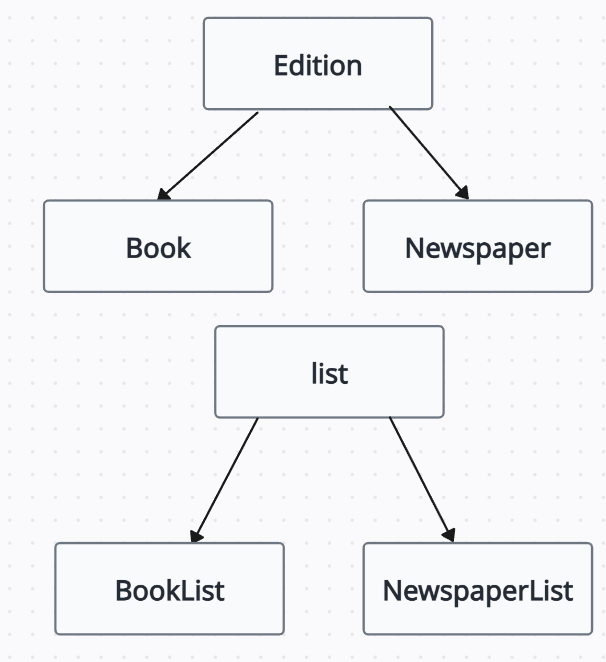


Рисунок 1 – Иерархия классов

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

Переопределены два метода класса object:

\_\_str\_\_ в Book и Newspaper: Этот метод определяет, как объект будет преобразован в строку, то есть что будет показано при попытке распечатать объект с помощью функции print() или просто путем вызова str() на объекте.

\_\_eq\_\_ в Book и Newspaper: Этот метод используется для сравнения двух объектов на эквивалентность. По умолчанию, он только сравнивает, являются ли объекты одним и тем же экземпляром. Переопределение этого метода позволяет изменить это поведение для сравнения объектов основываясь на их содержимом или некоторых их атрибутах.

3. В каких случаях будут использованы методы \_\_str\_\_() и \_\_eq\_\_():

Метод \_\_str\_\_() будет использоваться всегда, когда Python пытается преобразовать объект вашего класса в строку. Это может быть при вызове функции print() с объектом в качестве аргумента, или когда вы пытаетесь использовать оператор приведения к строке str() с объектом вашего класса. Этот метод позволяет более читабельное представление объекта для человека.

Метод \_\_eq\_\_() вызывается, когда вы пытаетесь сравнить два объекта с помощью оператора “==”. Это позволяет определить, когда считать объекты равными основываясь на их содержимом или атрибутах, а не их идентификаторах.

4. Переопределенные методы для BookList и NewspaperList будут работать, потому что они наследуются от стандартного класса list и поведение методов изменяется так, чтобы они работали специально для Book и Newspaper соответственно.

В BookList, переопределение метода append() делает возможным добавлять в список только объекты типа Book. Это предполагает что, если пользователь попытается добавить некорректные данные, например объект Newspaper или любой другой тип, вызовется исключение TypeError.

Пример:

book\_list = BookList("Favorites")

book1 = Book("BookName", 100, 16, "b", "Author Name", True, 256)

book\_list.append(book1) *#Это сработает, потому что book - это экземпляр Book*

*#Если попытаться добавить не Book объект, будет ошибка:*

newspaper1 = Newspaper("NewsName", 10, 1, "c", True, "Country", 7)

book\_list.append(newspaper1) *#вызовет TypeError*

В NewspaperList, метод extend() был специально изменен, чтобы обеспечить добавление только объектов типа Newspaper. Поведение отличается от стандартного extend, так как добавляет дополнительную проверку типа объектов в итерируемом аргументе.

Пример:

newspaper\_list = NewspaperList("GlobalNews")

newspaper1 = Newspaper("NewsName", 10, 1, "c", True, "Country", 7)

*#В этом случае можно добавить элементы типа Newspaper через extend*

newspaper\_list.extend([newspaper1])

*#Если попытаться добавить что-то иное, добавления не произойдет, поскольку extend проверяет тип*

book1 = Book("BookName", 100, 16, "b", "Author Name", True, 256)

newspaper\_list.extend([book1])

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | book1 = Book("Война и мир", 500, 12, "b", "Толстой", True, 1225)  book2 = Book("Евгейний Онегин", 300, 11, "c", "Пушкин", False, 1333)  newspaper1 = Newspaper("Times", 10, 1, "c", True, "UK", 7)  newspaper2 = Newspaper("Times", 11, 20, "b", False, "UK", 8)  my\_books = BookList("Моя библиотека")  my\_books.append(book1)  my\_books.append(book2)  my\_newspapers = NewspaperList("Мои газеты")  my\_newspapers.extend([newspaper1, newspaper2])  print(book1.\_\_str\_\_())  print(newspaper1.\_\_str\_\_())  print(book1.\_\_eq\_\_(book2))  print(newspaper1.\_\_eq\_\_(newspaper2))  print(my\_books.total\_pages())  my\_newspapers.print\_total\_price() | Book: название Война и мир, цена 500, возрастное ограничение 12, стиль b, автор Толстой, твердый переплет True, количество страниц 1225.  Newspaper: название Times, цена 10, возрастное ограничение 1, стиль c, интернет издание True, страна UK, периодичность 7.  False  True  2558  21 | Создается элементы book1 и book2 класса Book и элементы newspaper1 и newspaper2 класса Newspaper. Первая пара элементов добавляется в список my\_books, вторая – my\_newspapers. Применяется метод \_\_str\_\_ к book1 и newspaper1, затем применяется метод \_\_eq\_\_ для сравнения book1 и book2, затем newspaper1 и newspaper2, выводится общее количество страниц во всех книгах, а затем суммарная цена всех газет. |
|  | fake\_book = Newspaper(“Fake News", 5, 1, “c", False, “Nowhere", 1)  my\_books = BookList("Псевдо библиотека")  my\_books.append(fake\_book) | TypeError: Invalid type <class '\_\_main\_\_.Newspaper'> | При попытке добавить в список книг элемента, не принадлежащего классу Books, выводится сообщение об ошибке. |
|  | invalid\_book = Book("Странный роман", 300, 6, "b", "Я", False, -100) | ValueError: Invalid value | При попытке создания элемента класса Book значение атрибута pages отрицательное, что неверно по условию, выводится сообщение об ошибке. |
|  | fake\_news = "a"  my\_newspapers = NewspaperList("Моя газета")  my\_newspapers.extend([fake\_news])  print(len(my\_newspapers)) | 0 | Попытка добавить в список элемента, не принадлежащего к классу Newspaper, закончилась неудачей, длина списка равна нулю. |

## Выводы

В ходе работы был изучен теоретический материал по парадигмам программирования.

Изучена детально реализация функционального программирования на Python, усвоенные навыки применены на практике.

Изучены принципы объектно-ориентированной парадигмы программирования.

С применением полученных знаний реализована программа, которая хранит пользовательские данные и использует классы и модифицированные методы базового класса для работы с данными. В программе также применяется наследование для построения моделей данных.

# 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 in ("b", "c")):

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):

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 element in self:

total += element.pages

return total

def print\_count(self):

count = 0

for element in self:

count += 1

print(count)

class NewspaperList(list):

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

self.name = name

def extend(self, iterable):

for element in iterable:

if isinstance(element, Newspaper):

self.append(element)

def print\_age(self):

min\_age = 100

for element in self:

if element.age\_limit < min\_age:

min\_age = element.age\_limit

print(min\_age)

def print\_total\_price(self):

total = 0

for element in self:

total += element.price

print(total)