**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 |  | Шуменков А.П. |
| 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. В строке class Edition: определяется класс Edition, который является базовым для всех изданий. Метод \_\_init\_\_ определен, чтобы инициализировать объекты класса Edition с четырьмя параметрами: name, price, age\_limit, style. Проверка в условии if not all([...]) проверяет корректность переданных значений для инициализации объекта. Для успешной инициализации объекта значения должны соответствовать следующим условиям:

- name должно быть строкой (str),

- price должно быть целым числом (int) больше нуля,

- age\_limit должно быть целым числом (int) больше нуля,

- style должно быть одним из двух символов: "c" или "b".

Если хотя бы одно из условий не выполнено, выбрасывается исключение ValueError с сообщением "Invalid value". Если значения прошли проверку, то они присваиваются соответствующим атрибутам объекта: self.name, self.price, self.age\_limit, self.style.

2. Создается класс Book, который является подклассом класса Edition. Это означает, что класс Book наследует все атрибуты и методы класса Edition, а также может добавлять свои специфичные атрибуты и методы. Определяется метод \_\_init\_\_ в классе Book, который принимает 7 параметров: name, price, age\_limit, style, author, hardcover, pages. С помощью super().init(name, price, age\_limit, style) вызывается метод инициализации родительского класса Edition для инициализации базовых атрибутов name, price, age\_limit, style. В блоке if isinstance(...) происходит проверка корректности аргументов, переданных для инициализации объекта класса Book.

- author должно быть строкой (str),

- hardcover должно быть логическим значением (bool),

- pages должно быть целым числом (int) больше нуля.

Если хотя бы одно из условий не выполнено, выбрасывается исключение ValueError с сообщением "Invalid value". Если все проверки пройдены успешно, атрибуты author, hardcover, pages присваиваются объекту Book.

3. Метод str(self):

- Этот метод представляет собой специальный метод Python, который возвращает строковое представление объекта. В данном случае, метод str возвращает информацию об объекте класса Book в формате строки.

- Возвращаемая строка содержит данные об атрибутах объекта Book, таких как name, price, age\_limit, style, author, hardcover, pages.

- С помощью метода f-string строковое представление объекта формируется на основе значений этих атрибутов.

- Например, если у нас есть объект my\_book типа Book, то вызов print(my\_book) будет выводить строку вида:

"Book: название name, цена price, возрастное ограничение age\_limit, стиль style, автор author, твердый переплет hardcover, количество страниц pages."

4. Метод eq(self, other):

- Этот метод определяет логику сравнения объектов класса Book.

- Принимает два параметра: self (текущий объект) и other (другой объект, с которым сравниваем текущий).

- В данном случае, метод сравнивает значения атрибутов author и name текущего объекта и другого объекта (переданного как параметр).

- Если значения author и name равны у обоих объектов, метод возвращает True, иначе - False.

- Этот метод позволяет определить условие равенства двух объектов на основе значений их атрибутов author и name.

Таким образом, методы str и eq позволяют формировать строковое представление объекта класса Book и определять логику сравнения двух объектов класса Book на основе их атрибутов author и name.

5. Класс Newspaper наследует класс Edition, что означает, что класс Newspaper имеет все атрибуты и методы класса Edition. Метод init(self, name, price, age\_limit, style, online\_edition, country, frequency):

- Этот метод инициализирует объект класса Newspaper с определенными атрибутами и проверяет их корректность.

- Сначала вызывается метод init родительского класса Edition с помощью super().init(name, price, age\_limit, style), чтобы инициализировать базовые атрибуты.

- После этого идет проверка на корректность значений атрибутов online\_edition, country и frequency:

- online\_edition должен быть булевого типа (True или False).

- country должен быть строкой.

- frequency должен быть целым числом больше нуля.

- Если значения соответствуют данным условиям, они присваиваются соответствующим атрибутам объекта.

- В противном случае, вызывается исключение ValueError с сообщением "Invalid value". Метод str(self):

- Этот метод возвращает строковое представление объекта класса Newspaper.

- Строка формируется с использованием значений атрибутов объекта, включая базовые и добавленные в методе init.

- Возвращаемая строка содержит информацию о названии, цене, возрастном ограничении, стиле, наличии интернет издания, стране и периодичности газеты. Метод eq(self, other):

- Этот метод определяет условие равенства между объектами класса Newspaper.

- Метод сравнивает значения атрибутов name и country текущего объекта с другим объектом (переданным как параметр).

- Если значения этих атрибутов равны у обоих объектов, то метод возвращает True, иначе - False.

6. Класс BookList:

Метод init(self, name):

- Этот метод инициализирует объект класса BookList с атрибутом name.

- Используется метод super().init(self) для инициализации самого списка.

- Атрибут name устанавливается в значение, переданное в качестве параметра. Метод append(self, p\_object):

- Этот метод добавляет объект p\_object в список, если он является экземпляром класса Book.

- Если объект не является книгой, генерируется исключение TypeError с сообщением о неверном типе объекта. Метод total\_pages(self):

- Этот метод возвращает суммарное количество страниц всех книг в списке.

- Используется генераторное выражение для суммирования числа страниц каждой книги. Метод print\_count(self):

- Этот метод выводит количество книг в списке с помощью функции print и функции len

7. Класс NewspaperList:

Метод init(self, name):

- Этот метод инициализирует объект класса NewspaperList с атрибутом name.

- Используется метод super().init() для инициализации самого списка.

- Атрибут name устанавливается в значение, переданное в качестве параметра. Метод extend(self, iterable):

- Этот метод добавляет в список только те элементы из iterable, которые являются экземплярами класса Newspaper.

- Для фильтрации используется lambda-выражение в сочетании с функцией filter. Метод print\_age(self):

- Этот метод выводит минимальный возрастной рейтинг, присвоенный газетам в списке.Метод print\_total\_price(self):

- Этот метод выводит общую цену всех газет в списке, используя генераторное выражение для суммирования цен объектов.

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 |
|  | 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 |
|  | 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 |
|  | books = BookList('books')  book1 = Book('Name1', 90, 12, 'b', 'Ivanov', True, 250)  book2 = Book('Name2', 90, 12, 'b', 'Ivanov', True, 130)  books.append(book1)  books.append(book2)  print(books.total\_pages())  books.print\_count() | 380  2 | Проверка методов класса BookList |
|  | 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

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 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 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)

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.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]))