**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. 3341 |  | Кузнецова С.Е. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Цель – рассмотреть понятия парадигм программирования и освоить некоторые из них на практике.

Задачи:

1. Рассмотреть понятия парадигм программирования

2. Подробнее рассмотреть реализацию функционального программирования на Python с решением задач на практике.

3. Написать программу с использованием классов для представления книг и газет, а также их списков.

4. Переопределить методы базового класса

## 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

Термин «парадигма программирования» имеет множество определений, но в общем его можно описать так: парадигма программирования – это подход к программированию, описанный совокупностью идей и понятий, определяющих стиль написания компьютерных программ.

Python является объектно-ориентированным языком, но при этом на Python можно писать программы в процедурном стиле. Тем не менее, все, с чем приходится сталкиваться, даже используя процедурный стиль, является объектом: модули, функции, списки, строки и т. д. Любая программа на языке Python представляет собой совокупность объектов.

Как уже отмечалось, объект – конкретная сущность предметной области, тогда как класс – это тип объекта.

Классы содержат атрибуты, которые подразделяются на поля и методы.

Под методом понимают функцию, которая определена внутри класса. Поле – это переменная, которая определена внутри класса.

Объектно-ориентированная парадигма базируется на нескольких принципах: наследование, инкапсуляция, полиморфизм.

Наследование – специальный механизм, при котором можно расширять классы, усложняя их функциональность. Наследование позволяет повторно использовать функциональность базового класса, не меняя при этом базовый класс, а также расширять ее, добавляя новые атрибуты.

Под инкапсуляцией часто понимают сокрытие внутренней реализации от пользователя. В других языках программирования это достигается использованием модификаторов доступа; таким образом, в описании класса можно указать, какой атрибут будет доступен извне, а какой – нет.

Полиморфизм (polymorphism) (от греческого polymorphos) — это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач. Целью полиморфизма, применительно к объектно-ориентированному программированию, является использование одного имени для задания общих для класса действий. Рассуждая о полиморфизме в контексте ООП, обычно говорят о переопределении методов.

Исключения – это специальный класс объектов в языке Python. Они предназначены для управления поведением программы при возникновении ошибки, или, другими словами, для управления теми участками программного кода, где может возникнуть ошибка.

## Выполнение работы

1. Создаем класс Edition, который содержит атрибуты name, price, age\_limit, style. При инициализации объекта проверяем, что style является "c" или "b", а price, age\_limit больше нуля и типа int, а name типа str.

2. Создаем класс Book, который наследуется от класса Edition и добавляет атрибуты author, hardcover, pages. При инициализации объекта вызываем init родительского класса, затем проверяем, что author – объект типа str, hardcover – bool, pages – int и больше нуля. Переопределяем методы str и eq. Метод str возвращает строку в определённом виде, eq сравнивает два значения класса Book.

3. Создаем класс Newspaper, который также наследуется от класса Edition и добавляет атрибуты online\_edition, country, frequency. При инициализации объекта вызываем init родительского класса, затем проверяем, что online\_edition – объект типа bool, country – str, frequency – int и больше нуля. Переопределяем методы str и eq. Метод str возвращает строку в определённом виде, eq сравнивает два значения класса Newapaper.

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

5. Создаем класс NewspaperList, аналогично BookList, 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 реализует иерархию классов изданий и списков для хранения изданий каждого класса. Каждый класс издания имеет свои уникальные атрибуты и методы.

1. Изображение иерархии классов:

```

Edition -> Book

Edition -> Newspaper

List -> BookList

List -> NewspaperList

```

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

Метод \_\_init\_\_(): переопределен в каждом классе для инициализации атрибутов.

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() списка.

Метод extend(iterable): Переопределение метода extend() 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 созданных списков будут работать, т.к. мы сами задали необходимые условия для их работы в классах BookList и NewspaperList. Не переопределённые методы будут работать без учёта специфики классов элементов.

Пример:

```

lst = BookList('booklist')

lst.append(Book('Dune', 303, 16, 'c', 'Herbert Frank', True, 768)) # *добавление книги*

lst.extend([0,1,2]) # *добавятся элементы 0, 1, 2, не являющиеся книгами*

lst.print\_count() *# 4 элемента в списке*

print(a[0], a[1], a[2], a[3]) # *Book: название Dune, цена 303, возрастное ограничение 16, стиль c, автор Herbert Frank, твердый переплет True, количество страниц 768. 0 1 2*

```

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | books = BookList('booklist')  book1 = Book('Dune', 303, 16, 'c', 'Herbert Frank', True, 768)  book2 = Book('Alice in Wonderland', 377, 12, 'c', 'Lewis Carroll', True, 160)  books.append(book1)  books.append(book2)  print(books.total\_pages())  books.print\_count()  print(book1.\_\_str\_\_())  print(book1.\_\_eq\_\_(book2))  newspapers = NewspaperList('newspaperlist')  news1 = Newspaper('Animals', 35, 6, 'b', True, 'Russia', 100)  news2 = Newspaper('Politics', 120, 16, 'b', True, 'USA', 30)  newspapers.extend([news1, news2])  newspapers.print\_age()  newspapers.print\_total\_price()  print(news1.\_\_str\_\_())  print(news1.\_\_eq\_\_(news2)) | 928  2  Book: название Dune, цена 303, возрастное ограничение 16, стиль c, автор Herbert Frank, твердый переплет True, количество страниц 768.  False  6  155  Newspaper: название Animals, цена 35, возрастное ограничение 6, стиль b, интернет издание True, страна Russia, периодичность 100.  False | Проверка работы основных методов классов |
|  | books = BookList('booklist')  books.append(3) | TypeError: Invalid type <class 'int'> | Проверка неправильных данных |
|  | book1 = Book('Dune', 'me', 16, 'c', 'Herbert Frank', True, 768) | ValueError: Invalid value | Проверка неправильных данных |
|  | news1 = Newspaper('Animals', 35, 6, 'b', True, 10, 100) | ValueError: Invalid value | Проверка неправильных данных |
|  | newspapers = NewspaperList('newslist')  news1 = Newspaper('Animals', 35, 6, 'b', True, 'Russia', 100)  newspapers.extend([news1])  newspapers.extend([0,1,2])  newspapers.extend(['boop'])  print(len(newspapers)) | 1 | Проверка метода extend – добавляет в список только объекты класса Newspaper |

## Выводы

Рассмотрели понятие парадигм программирования.

Подробнее рассмотрели реализацию функционального программирования на Python с решением задач на практике.

Рассмотрели обработку исключительных ситуаций и способы их генерации на Python.

Изучили принципы объектно-ориентированной парадигмы программирования.

Изучив иерархию классов, поняли, как использовать наследование для создания классов с общими характеристиками, поддерживая при этом уникальные особенности и методы для каждого класса. Также рассмотрели, как переопределить методы базового класса object для более удобной работы с объектами и их строковым представлением.

# Приложение А Исходный код программы

Название файла: main.py

class Edition:

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

if not (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')):

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) and isinstance(hardcover, bool) and isinstance(pages, int) and 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.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 not (isinstance(online\_edition, bool) and isinstance(country, str) and isinstance(frequency, int) and frequency>0):

raise ValueError("Invalid value")

else:

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):

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):

temp = []

for object in iterable:

if isinstance(object, Newspaper):

temp.append(object)

super().extend(temp)

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]))