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 _____ Рябов М.Л.

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

1 вариант.

Даны фигуры в двумерном пространстве.

Базовый класс - фигура Figure:

class Figure:

- Поля объекта класса Figure:
- периметр фигуры (в сантиметрах, целое положительное число)
- площадь фигуры (в квадратных сантиметрах, целое положительное число)
- цвет фигуры (значение может быть одной из строк: 'r', 'b', 'g').

При создании экземпляра класса Figure необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

Многоугольник - Polygon:

class Polygon: #Наследуется от класса Figure

Поля объекта класса Polygon:

- периметр фигуры (в сантиметрах, целое положительное число)
- площадь фигуры (в квадратных сантиметрах, целое положительное число)
- цвет фигуры (значение может быть одной из строк: 'r', 'b', 'g')
- количество углов (неотрицательное значение, больше 2)
- равносторонний (значениями могут быть или True, или False)
- самый большой угол (или любого угла, если многоугольник равносторонний) (целое положительное число)

При создании экземпляра класса Polygon 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: Polygon: Периметр <периметр>, площадь <площадь>, цвет фигуры <цвет фигуры>, количество углов <кол-во углов>, равносторонний <равносторонний>, самый большой угол <самый большой угол>.

Метод __add__():

Сложение площади и периметра многоугольника. Возвращает число, полученное при сложении площади и периметра многоугольника.

Метод __eq__():

Метод возвращает True, если два объекта класса равны и False иначе. Два объекта типа Polygon равны, если равны их периметры, площади и количество углов.

Окружность - Circle:

class Circle: #Наследуется от класса Figure

Поля объекта класса Circle:

- периметр фигуры (в сантиметрах, целое положительное число)
- площадь фигуры (в квадратных сантиметрах, целое положительное число)
- цвет фигуры (значение может быть одной из строк: 'r', 'b', 'g').
- радиус (целое положительное число)
- диаметр (целое положительное число, равен двум радиусам)

При создании экземпляра класса Circle 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: Circle: Периметр <периметр>, площадь <площадь>, цвет фигуры <цвет фигуры>, радиус <радиус>, диаметр <диаметр>.

Метод __add__():

Сложение площади и периметра окружности. Возвращает число, полученное при сложении площади и периметра окружности.

Метод __eq__():

Метод возвращает True, если два объекта класса равны и False иначе. Два объекта типа Circle равны, если равны их радиусы.

Необходимо определить список list для работы с фигурами:

Многоугольники:

class PolygonList – список многоугольников - 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 - многоугольник (объект класса Polygon), элемент добавляется в список, иначе выбрасывается исключение TypeError с текстом: Invalid type <тип_объекта p_object>

Метод print_colors(): Вывести цвета всех многоугольников в виде строки (нумерация начинается с 1):

<i> многоугольник: <color[i]>

<j> многоугольник: <color[j]> ...

Метод print_count(): Вывести количество многоугольников в списке.

Окружности:

class CircleList – список окружностей - 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 качестве аргумента передается итерируемый объект iterable, в случае, если элемент iterable - объект класса Circle, этот элемент добавляется в список, иначе не добавляется.

Метод print_colors(): Вывести цвета всех окружностей в виде строки (нумерация начинается с 1):

<i> окружность: <color[i]>

<j> окружность: <color[j]> ...

Метод total_area(): Посчитать и вывести общую площадь всех окружностей.

Основные теоретические положения

Классы в языке Python представляют собой механизм объектно-ориентированного программирования, который позволяет создавать новые типы данных, инкапсулируя в них данные и методы для их обработки. Классы состоят из атрибутов - переменных, хранящих данные объекта и методов - функций, обрабатывающих эти данные.

Для создания нового класса используется ключевое слово "class", за которым следует название класса и двоеточие. Объекты класса создаются с помощью оператора точки, что позволяет получить доступ к их методам и атрибутам. Конструктор класса, метод "init", используется для инициализации объекта и присвоения начальных значений его атрибутам.

Наследование позволяет создавать новые классы на основе уже существующих, наследуя их атрибуты и методы. Полиморфизм позволяет использовать объекты разных классов с одинаковым интерфейсом, упрощая код и уменьшая повторяемость. Инкапсуляция позволяет скрыть детали реализации класса от внешнего мира, делая его более надежным и безопасным. Для этого используются приватные атрибуты и методы, начинающиеся с двойного подчеркивания.

Выполнение работы

В данной работе описаны два основных класса: Polygon и Circle, наследующиеся от класса Figure, который описывает общие характеристики фигуры - периметр, площадь и цвет.

В классе Polygon добавлены дополнительные характеристики для многоугольников: количество углов, равносторонний ли он и самый большой угол.

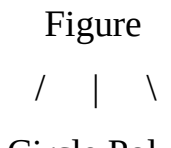
А в классе Circle - характеристики для окружности: радиус и диаметр.

Также описаны два класса-списка: PolygonList и CircleList, наследующиеся от списков. Они позволяют добавлять объекты только соответствующих классов и выводить информацию о цветах фигур.

В самом конце описан метод total_area(), который считает общую площадь всех окружностей в списке.

Этот код реализует иерархию классов фигур (окружность, многоугольник) и списков для хранения фигур каждого класса. Каждый класс фигуры имеет свои уникальные атрибуты и методы.

1. Изображение иерархии классов:



Circle Polygon

PolygonList <-- list

CircleList <-- list

2. В переопределении методов класса объекта object или других методов:

- Метод `__init__`: переопределен в каждом классе для инициализации атрибутов.

- Метод `__str__`: переопределен для возвращения строкового представления объекта.

3. Метод `__add__` в классе Polygon и в классе Circle переопределен таким образом, что значение area прибавляется к значению perimeter. Метод `__eq__` проверяет равенство атрибута area у двух объектов данного класса Polygon и атрибута radius у двух объектов класса Circle.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии

1.	fig = Figure(10,25,'g') #фигура
	print(fig.perimeter, fig.area, fig.color)
	
	polygon = Polygon(10,25,'g',4, True, 90) #многоугольник
	polygon2 = Polygon(10,25,'g',4, True, 90)
	print(polygon.perimeter, polygon.area, polygon.color, polygon.angle_count, polygon.equilateral, polygon.biggest_angle)
	print(polygon.__str__())
	print(polygon.__add__())
		10 25 g
		10 25 g 4 True 90
		Polygon: Периметр 10, площадь 25, цвет фигуры g, количество углов 4, равносторонний True, самый большой угол 90.
		35
		True
		13 13 r 2 4
		Circle: Периметр 13, площадь 13, цвет фигуры r, радиус 2, диаметр 4.
		26
		True
		1 многоугольник: g
		2 многоугольник: g
		2
		1 окружность: r
		2 окружность: g
		26
			Основной тест с сайта moevm

	print(polygon.__eq__(polygon2))
	
	circle = Circle(13, 13,'r', 2, 4)
	#окружность
	circle2 = Circle(13, 13,'g', 2, 4)
	print(circle.perimeter, circle.area, circle.color, circle.radius, circle.diametr)
	print(circle.__str__())
	print(circle.__add__())
	print(circle.__eq__(circle2))
	
	polygon_list = PolygonList(Polygon)
	#список многоугольников
	polygon_list.append(polygon)
	polygon_list.append(polygon2)
	polygon_list.print_colors()
	polygon_list.print_count()		

	circle_list = CircleList(Circle) #список окружностей circle_list.extend([circle, circle2]) circle_list.print_colors() circle_list.total_area()		

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 Figure:
    def __init__(self, perimeter, area, color):
          condition_per = perimeter > 0 and isinstance(perimeter,
int)
        condition_sqr = area > 0 and isinstance(area, int)
        condition_clr = color in ('r', 'b', 'g')
        if condition_per and condition_sqr and condition_clr:
            self.area = area
            self.perimeter = perimeter
            self.color = color
        else:

            raise ValueError("Invalid value")


class Polygon(Figure):
        def __init__(self, perimeter, area, color, angle_count,
equilateral, biggest_angle):
        super().__init__(perimeter, area, color)
        angle_cond = angle_count > 2
        cond_eq = isinstance(equilateral, bool)
        cond_big = isinstance(biggest_angle, int) and biggest_angle
> 0
        if angle_cond and cond_eq and cond_big:
            self.angle_count = angle_count
            self.equilateral = equilateral
            self.biggest_angle = biggest_angle
        else:
            raise ValueError("Invalid value")

    def __str__(self):
            return (f"Polygon: Периметр {self.perimeter}, площадь
{self.area}, цвет фигуры {self.color}, количество углов "
                                    f"{self.angle_count}, равносторонний
{self.equilateral}, самый большой угол {self.biggest_angle}.")

    def __add__(self):
        return self.area + self.perimeter

    def __eq__(self, other):
            return True if self.perimeter == other.perimeter and
self.area == other.area and self.angle_count == other.angle_count else
False


class Circle(Figure):
    def __init__(self, perimeter, square, color, radius, diametr):
        super().__init__(perimeter, square, color)
                if isinstance(radius, int) and radius > 0 and
isinstance(diametr, int) and diametr/radius == 2:
```

```
                self.radius = radius
                self.diametr = diametr
            else:
                raise ValueError("Invalid value")

        def __str__(self):
                return f"Circle: Периметр {self.perimeter}, площадь
{self.area}, цвет фигуры {self.color}, радиус {self.radius}, диаметр
{self.diametr}."

        def __add__(self):
            return self.area + self.perimeter

        def __eq__(self, other):
            return True if self.radius == other.radius else False


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

        def append(self, __object):
            if isinstance(__object, Polygon):
                super().append(__object)
            else:
                raise TypeError("Invalid type <тип_объекта p_object>")

        def print_colors(self):
            for i in range(0, super().__len__()):
                                        print(f"{i+1}   многоугольник:
{super().__getitem__(i).color}")

        def print_count(self):
            print(super().__len__())


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

        def extend(self, __iterable):
            for __obj in __iterable:
                if isinstance(__obj, Circle):
                    super().append(__obj)

        def print_colors(self):
            for i in range(0, super().__len__()):
                                            print(f"{i+1}   окружность:
{super().__getitem__(i).color}")

        def total_area(self):
            total = 0
            for i in range(0, super().__len__()):
                total += super().__getitem__(i).area
            print(total)
```