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. 3344 _____ Щербак М.С.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Получить базовые навыки работы с ООП на языке Python.

Задание

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.

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.

Выполнение работы

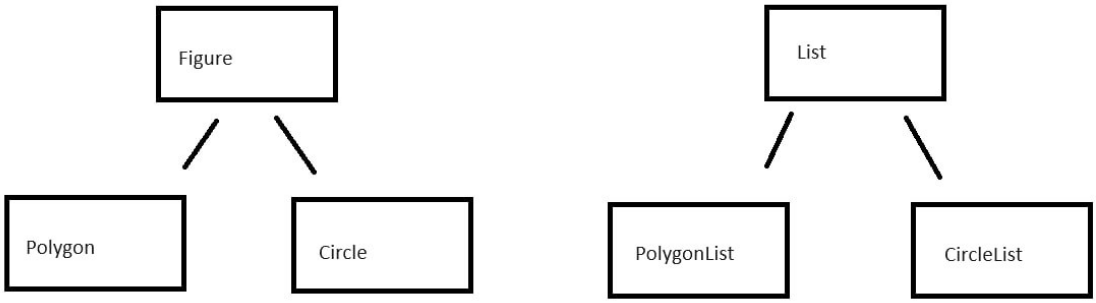

Рисунок 1 – Изображение иерархии классов

2. Методы, которые были переопределены в данном коде:

- В классе Figure:

- __init__: переопределен для инициализации объектов класса Figure с заданными параметрами perimeter, area, и color.

- add: добавляет периметр и площадь фигуры.

- В классе Polygon:

- __init__: переопределен для инициализации объектов класса Polygon с дополнительными параметрами, такими как angle_count, equilateral, и biggest_angle.

- __str__: возвращает строковое представление объекта Polygon с указанием его характеристик.

- eq: проверяет эквивалентность двух объектов типа Polygon.

- В классе Circle:

- __init__: переопределен для инициализации объектов класса Circle с дополнительными параметрами, такими как radius и diametr.

- __str__: возвращает строковое представление объекта Circle с указанием его характеристик.

- eq: проверяет эквивалентность двух объектов типа Circle.

- В классе PolygonList:

- __init__: переопределен для инициализации объектов класса PolygonList.

- append: добавляет объекты типа Polygon в список.

- print_colors: выводит цвета всех многоугольников в списке.

- print_count: выводит количество объектов типа Polygon в списке.

- В классе CircleList:

- __init__: переопределен для инициализации объектов класса CircleList.

- extend: расширяет список объектами типа Circle.

- print_colors: выводит цвета всех окружностей в списке.

- total_area: вычисляет и выводит общую площадь всех окружностей в списке.

3. Методы str() и add() будут использованы следующим образом:

- Метод str() вызывается при попытке получить строковое представление объекта (например, при вызове функции print()).

- Метод add() используется для вычисления суммы периметра и площади фигуры.

4. Переопределенные методы класса list для PolygonList и CircleList будут работать, поскольку оба класса наследуются от встроенного класса list. Примеры использования:

```
# Пример использования методов для PolygonList
poly_list = PolygonList("Polygon List")
poly1 = Polygon(10, 20, 'r', 4, True, 90)
poly_list.append(poly1)
```

```
poly_list.print_colors()
poly_list.print_count()

# Пример использования методов для CircleList
circle_list = CircleList("Circle List")
circle1 = Circle(15, 100, 'b', 5, 10)
circle_list.extend([circle1])
circle_list.print_colors()
circle_list.total_area()
```

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__())
	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,
		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
			Корректно

	circle2]) circle_list.print_colors() circle_list.total_area()		

Выводы

Были получены базовые навыки работы с ООП. С использованием ООП была написана программа для работы с информацией о печатных изданиях. Также были изучены наследование классов и переопределение методов.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.py

```
class Figure:
    '''Поля объекта класс Figure:
    perimeter - периметр фигуры (в сантиметрах, целое положительное число)
        area - площадь фигуры (в квадратных сантиметрах, целое положительное число)
     color - 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'.
     '''
    def __init__(self, perimeter, area, color):
        self.check_stats(perimeter, area, color)
        self.perimeter = perimeter
        self.area = area
        self.color = color

    def check_stats(self, perimeter, area, color):
        if not isinstance(perimeter, int) or perimeter <= 0:
            raise ValueError('Invalid value')
        if not isinstance(area, int) or area <= 0:
            raise ValueError('Invalid value')
        if color not in ['r', 'g', 'b']:
            raise ValueError('Invalid value')

    def __add__(self):
      # '''Сложение площади и периметра многоугольника. Возвращает число, полученное при сложении площади и периметра многоугольника.'''
        return self.perimeter + self.area


class Polygon(Figure):  # Наследуется от класса Figure
    '''Поля объекта класс Polygon:
    perimeter - периметр фигуры (в сантиметрах, целое положительное число)
        area - площадь фигуры (в квадратных сантиметрах, целое положительное число)
     color - цвет фигуры (значение может быть одной из строк: 'r', 'b', 'g')
      angle_count - количество углов (целое положительное значение, больше 2)
     equilateral - равносторонний (значениями могут быть или True, или False)
       biggest_angle - самый большой угол (или любой угол, если многоугольник равносторонний) (в градусах, 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'.
     '''
```

```
            def __init__(self, perimeter, area, color, angle_count,
equilateral, biggest_angle):
                self.check_stats(perimeter, area, color)
                super().__init__(perimeter, area, color)
                self.check_extra(angle_count, equilateral, biggest_angle)
                self.angle_count = angle_count
                self.equilateral = equilateral
                self.biggest_angle = biggest_angle

            def check_extra(self, angles, equlateral, biggest_angle):
                if not isinstance(angles, int) or angles < 3:
                    raise ValueError('Invalid value')
                if not isinstance(equlateral, bool):
                    raise ValueError('Invalid value')
                 if not isinstance(biggest_angle, int) or biggest_angle <=
0:
                    raise ValueError('Invalid value')

            def __str__(self):
                 #  '''Преобразование к строке вида: Polygon: Периметр
<периметр>, площадь <площадь>, цвет фигуры <цвет фигуры>, равносторонний
<равносторонний>, прямоугольный <прямоугольный>.'''
                  return f"Polygon: Периметр {self.perimeter}, площадь
{self.area},    цвет    фигуры    {self.color},    количество    углов
{self.angle_count}, равносторонний {self.equilateral}, самый большой угол
{self.biggest_angle}."

            def __eq__(self, other):
                 '''Метод возвращает True, если два объекта класса равны и
False иначе. Два объекта типа Polygon равны, если равны их периметр,
площадь и количество углов.'''
                if not other.__class__.__name__ == self.__class__.__name__:
                    return False
                 return self.perimeter == other.perimeter and self.area ==
other.area and self.angle_count == other.angle_count


        class Circle(Figure):  # Наследуется от класса Figure
            '''Поля объекта класс Circle:
            perimeter - периметр фигуры (в сантиметрах, целое положительное
число)
                area - площадь фигуры (в квадратных сантиметрах, целое
положительное число)
             color - цвет фигуры (значение может быть одной из строк: 'r',
'b', 'g')
            radius - радиус (целое положительное число)
               diametr - 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'.
            '''

            def __init__(self, perimeter, area, color, radius, diametr):
                self.check_stats(perimeter, area, color)
                super().__init__(perimeter, area, color)
                self.check_extra(radius, diametr)
```

```
            self.radius = radius
            self.diametr = diametr

        def check_extra(self, radius, diametr):
             if not isinstance(radius, int) or not isinstance(diametr, int) or not radius > 0 or not diametr == 2 * radius:
                raise ValueError('Invalid value')

        def __str__(self):
        # '''Преобразование к строке вида: Circle: Периметр <периметр>, площадь <площадь>, цвет фигуры <цвет фигуры>, радиус <радиус>, диаметр <диаметр>.'''
                  return f"Circle: Периметр {self.perimeter}, площадь {self.area}, цвет фигуры {self.color}, радиус {self.radius}, диаметр {self.diametr}."

        def __eq__(self, other):
         # '''Метод возвращает True, если два объекта класса равны и False иначе. Два объекта типа Circle равны, если равны их радиусы.'''
            if not other.__class__ == self.__class__:
                return False
            return self.radius == other.radius



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

        def append(self, p_object):
            if isinstance(p_object, Polygon):
                super().append(p_object)
            else:
                                    raise TypeError(f'Invalid type {type(p_object).__name__}')

        def print_colors(self):
        # '''Вывести цвета всех многоугольников.'''
            for i in range(len(list(self))):
                print(f"{i+1} многоугольник: {list(self)[i].color}")

        def print_count(self):
                    count = len([polygon for polygon in self if isinstance(polygon, Polygon)])
            print(count)

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

        def extend(self, iterable):
            for item in iterable:
                if isinstance(item, Circle):
                    self.append(item)
```

```
        def print_colors(self):
        # '''Вывести цвета всех изогнутых фигур.'''
            for i in range(len(list(self))):
                print(f"{i+1} окружность: {list(self)[i].color}")

        def total_area(self):
                total_area = sum([circle.area for circle in self if
isinstance(circle, Circle)])
            print(total_area)
```