**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.Вариант 1

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Коршков А.А. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Научится работать с классами, создавать методы и функции для классов, понять принцип наследования и переопределения, понять, как работает super().

Необходимо создать программу, которая может на основе различных классов создавать объекты фигур и работать с ними. Также программа должна уметь определять верный тип данных, а также уметь добавлять в определённую группу объектов.

## 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(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(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(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(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

В лабораторной работе необходимо создать классы с определёнными методами, которые представляют собой фигуры с определёнными параметрами и списки для хранения этих фигур.

Класс Figure будет родительским для классов Polygon и Circle и содержать в себе информацию о периметре, площади, цвете фигуры. При создании класса происходит проверка на тип входных данных, а также на их корректность (например, периметр и площадь должен быть положительными числами, а цвет фигуры – только одна из букв ‘r’ ‘g’ ‘b’).

Класс Polygon описывает многоугольник, поэтому у него в параметрах есть информация о количестве углов, равносторонний это многоугольник или нет, его самый большой угол. Был добавлен метод для вычисления суммы периметра и площади фигуры, метод для выводе информации об объекте, а также метод для сравнения двух объектов этого класса по периметру, площади и количеству углов. Проверяется, что количество углов больше 2, а больший угол больше нуля.

Класс Circle описывает окружность, с определённым радиусом, и диаметром. Были добавлены методы для вывода информации об объекте, метод для сравнения объектов по радиусам и метод для суммы периметра и площади окружности. Проверяется, что радиус и диаметр положительные числа, а диаметр равен 2 радиусам.

Метод \_\_str\_\_() отвечает за строковое представление объекта, т.е. когда происходит преобразование объекта класса в тип str.

Метод \_\_add\_\_() отвечает за сложение двух объектов класса (в нашем случае происходит сложение площади и периметра одной фигуры).

Класс PolygonList наследуется от класса list. Был переопределён append, при котором осуществляется проверка осуществляется, что добавляемый элемент является объектом класса Polygon. Метод print\_colors выводит информацию о цвете каждого многоугольника, а print\_counts – кол-во элементов в списке.

Класс CircleList наследуется от класса list. Был переопределён extend, ему передаётся итерируемый объект, из которого нужно добавить только те элементы, которые являются объектами класса Circle. Метод print\_colors выводит информацию о цвете каждого многоугольника, а total\_area – общую площадь всех окружностей.

Переопределённые методы append и extend в классах PolygonList и CircleList работают корректно за счёт использования super(). Благодаря нему можно обратиться к методам append и extend из списка и корректно добавить объект.

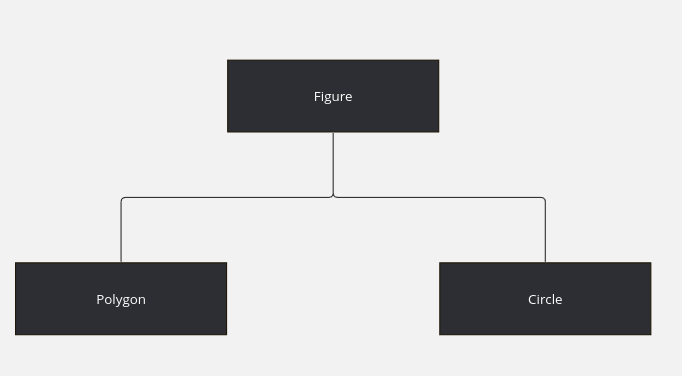


Рисунок 1 – Иерархия классов фигур

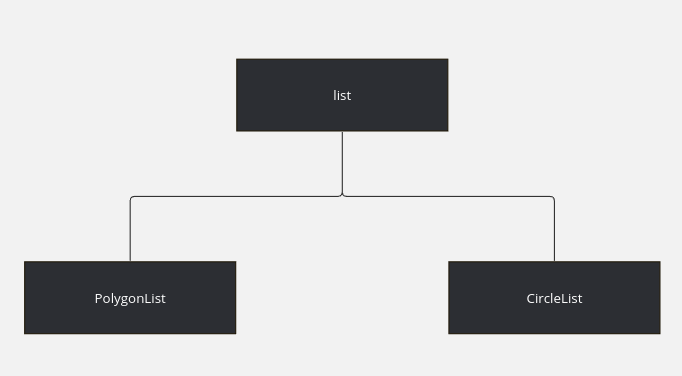


Рисунок 2 – Иерархия классов списков фигур

## Выводы

Были изучены способы наследования от различных классов, переопределение методов, а также возможность использовать функции super() для того, чтобы использовать методы родительского класса.

Была реализована программа, которая создаёт объекты классов различных фигур, может добавлять их в определённые группы и работать с ними.

# Приложение А Исходный код программы

Название файла: main.py

class Figure:

def \_\_init\_\_(self, perimeter, area, color):

if type(perimeter) is not int or type(area) is not int or type(color) is not str:

raise ValueError("Invalid value")

if (perimeter <= 0) or (area <= 0) or color not in ("r", "g", "b"):

raise ValueError("Invalid value")

self.perimeter = perimeter

self.area = area

self.color = color

class Polygon(Figure):

def \_\_init\_\_(self, perimeter, area, color, angle\_count, equilateral, biggest\_angle):

super().\_\_init\_\_(perimeter, area, color)

if type(angle\_count) is not int or type(equilateral) is not bool or type(biggest\_angle) is not int:

raise ValueError("Invalid value")

if (angle\_count < 2) or (angle\_count <= 0) or (biggest\_angle <= 0):

raise ValueError("Invalid value")

self.angle\_count = angle\_count

self.equilateral = equilateral

self.biggest\_angle = biggest\_angle

def \_\_str\_\_(self):

return (f"Polygon: Периметр {self.perimeter}, площадь {self.area}, цвет фигуры {self.color}, "

f"количество углов {self.angle\_count}, равносторонний {self.equilateral}, "

f"самый большой угол {self.biggest\_angle}.")

def \_\_add\_\_(self):

return self.perimeter + self.area

def \_\_eq\_\_(self, other):

return self.perimeter == other.perimeter and self.area == other.area and self.angle\_count == other.angle\_count

class Circle(Figure): # Наследуется от класса Figure

def \_\_init\_\_(self, perimeter, area, color, radius, diametr):

super().\_\_init\_\_(perimeter, area, color)

if type(radius) is not int or type(diametr) is not int:

raise ValueError("Invalid value")

if radius <= 0 or diametr != 2 \* radius:

raise ValueError("Invalid value")

self.radius = radius

self.diametr = diametr

def \_\_str\_\_(self):

return (f"Circle: Периметр {self.perimeter}, площадь {self.area}, "

f"цвет фигуры {self.color}, радиус {self.radius}, диаметр {self.diametr}.")

def \_\_add\_\_(self):

return self.perimeter + self.area

def \_\_eq\_\_(self, other):

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)}")

def print\_colors(self):

for i in range(len(self)):

print(f"{i + 1} многоугольник: {self[i].color}")

def print\_count(self):

print(len(self))

class CircleList(list):

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

super().\_\_init\_\_()

self.name = name

def extend(self, iterable):

for i in iterable:

if isinstance(i, Circle):

super().append(i)

def print\_colors(self):

for i in range(len(self)):

print(f"{i + 1} окружность: {self[i].color}")

def total\_area(self):

k = 0

for i in self:

k += i.area

print(k)

# Приложение Б тЕСТИРОВАНИЕ

Входные данные (программа):

print("Тест №1 (вызовы методов функции Polygon):")

polygon = Polygon(10, 25, 'g', 4, True, 90)

polygon2 = Polygon(10, 25, 'g', 4, True, 90)

print(polygon.\_\_str\_\_())

print(polygon.\_\_add\_\_())

print(polygon.\_\_eq\_\_(polygon2))

print()

print("Тест №2 (вызовы методов функции Circle):")

circle = Circle(13, 13, 'r', 2, 4)

circle2 = Circle(13, 13, 'g', 2, 4)

print(circle.\_\_str\_\_())

print(circle.\_\_add\_\_())

print(circle.\_\_eq\_\_(circle2))

print()

print("Тест №3 (создание списка для многоугольника, добавление объектов в список):")

polygon\_list = PolygonList(Polygon)

polygon\_list.append(polygon)

polygon\_list.append(polygon2)

polygon\_list.print\_colors()

print()

print("Тест №4 (создание списка для окружности, добавление набора элементов в список):")

circle\_list = CircleList(Circle)

circle\_list.extend([circle, circle2, polygon])

circle\_list.print\_colors()

print()

Выходные данные (вывод самой программы):

Тест №1 (вызовы методов функции Polygon):

Polygon: Периметр 10, площадь 25, цвет фигуры g, количество углов 4, равносторонний True, самый большой угол 90.

35

True

Тест №2 (вызовы методов функции Circle):

Circle: Периметр 13, площадь 13, цвет фигуры r, радиус 2, диаметр 4.

26

True

Тест №3 (создание списка для многоугольника, добавление объектов в список):

1 многоугольник: g

2 многоугольник: g

Тест №4 (создание списка для окружности, добавление набора элементов в список):

1 окружность: r

2 окружность: g

Комментарий:

Тест №3 и Тест №4 показывают, что переопределённые методы работают корректно. В последнем тесте в extend был добавлен объект polygon и из этого теста видно, что он не был добавлен в список circle\_list.