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. 3384 _____ Рудаков А.Л.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы.

Изучить теорию об основных парадигмах программирования. Применить знания на практике реализовав программный код реализации декларативного программирования 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.

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.

Выполнение работы.

Внутри программы создано 5 классов:

1: *oibject*

2: *Figure* *list*

3: *Circle* *Polygon* *CircleList* *PolygonList*

В начале программы был создан класс *Figure*, полями которого стали *perimetr, area* и *color*. Также внутри метода *__init__()*, в которой и инициализируются поля класса идет проверка на то, являются ли входные данные корректными, если это не так, то при помощи оператора *raise* выбрасывается ошибка *ValueError* с текстом *'Invlaid value'*.

Далее был создан класс *Polygon*, наследуемый от класса *Figure*, полями которого стали perimetr, *area, color, angle_count, equilateral* и *biggest_angle*. Также внутри метода *__init__()*, в которой и инициализируются поля класса идет проверка на то, являются ли входные данные корректными, если это не так, то при помощи оператора *raise* выбрасывается ошибка *ValueError* с текстом *'Invlaid value'*. Далее определен метод *__str__()*, в котором при помощи оператора *return* возвращается строка с информацией. Кроме этого определен метод *__add__()*, который возвращает сумму периметра и площади фигуры. Последним в данном классе определен метод *__eq__()*, который сравнивает два

объекта данного класса по площади, периметру и количеству углов, возвращая *True* или *False*.

Далее был создан класс *Circle*, наследуемый от класса *Figure*, полями которого стали perimetr, *area, color, radius, diametr*. Также внутри метода *__init__()*, в которой и инициализируются поля класса идет проверка на то, являются ли входные данные корректными, если это не так, то при помощи оператора *raise* выбрасывается ошибка *ValueError* с текстом *'Invlaid value'*. Далее определен метод *__str__()*, в котором при помощи оператора *return* возвращается строка с информацией. Кроме этого определен метод *__add__(),* который возвращает сумму периметра и площади фигуры. Последним в данном классе определен метод *__eq__()*, который сравнивает два объекта данного класса по площади, возвращая *True* или *False*.

Далее создан класс *PolygonList*, наследуемый от класса *list*, полем которого является *name*, инициализируемое в методе *__init__()*. Далее переопределен метод *append()*, в котором при помощи функции *isistence()* проверяется, является ли поданный объект, объектом класса *Circle*. Если да, то он добавляется в список, если нет, то при помощи оператора raise выбрасывается ошибка *TypeError* с текстом *f'Invalid type {type(p_object)}'*, где при помощи функции *type()* выводиться тип поданного объекта. Далее определен метод *print_colors()*, в котором происходит проход по списку и вывод цветов элементов списка. Последним реализован метод *print_count()*, который выводит длину списка.

Далее создан класс *CircleList*, наследуемый от класса *list*, полем которого является *name*, инициализируемое в методе *__init__()*. Далее переопределен метод *extend()*, в котором при помощи функции *isistence()* проверяется, является ли поданный объект, объектом класса *Polygon*. Если да, то он добавляется в список, если нет, то при помощи оператора raise выбрасывается ошибка *TypeError* с текстом *f'Invalid type {type(p_object)}',* где при помощи функции *type()* выводиться тип поданного объекта. Далее определен метод *print_colors(),* в котором происходит проход по списку и вывод цветов элементов списка.

Последним реализован метод *total_area()*, который сумму всех площадей элементов списка.

Подводя итог: в классах *Polygon* и *Circle* были переопределены методы *__str__()*, *__add__()* и *__eq__()*, который приходят в действие, когда их вызывают, в классе PolygonList переопределен метод *append()*, который корректно работает и добавляет в список только нужные элементы, иначе выкидывает ошибку, в классе *CircleList* переопределен метод *extend()*, который корректно работает и добавляет в список только те списки, элементы которого являются нужными, в противном случае выкидывает ошибку.

Разработанный программный код см. в приложении А.

Тестирование см. в приложении Б.

Выводы.

Была изучена теория о парадигмах программирования. Был создан программный код внутри которого были созданы новые классы Figure, Polygon, Circle, PolygonList, CircleList, внутри которых определены новые методы и переопределены старые.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.py

```
class Figure:
    def __init__(self,perimeter,area,color):
        if isinstance(perimeter,int) and perimeter>0 and isinstance(area,int) and area>0 and (color=='r' or color=='b' or color=='g'):
            self.perimeter = perimeter
            self.area = area
            self.color = color
        else:
            raise ValueError('Invlaid value')


class Polygon(Figure): #Наследуется от класса Figure
    def __init__(self,perimeter,area,color,angle_count,equilateral,biggest_angle):
        if isinstance(perimeter,int) and isinstance(area,int) and isinstance(angle_count,int) and isinstance(equilateral,bool) and isinstance(biggest_angle,int) and (equilateral==True or equilateral==False) and biggest_angle>0 and perimeter>0 and area>0 and angle_count>2 and (color=='r' or color=='b' or color=='g'):
            super().__init__(perimeter, area, color)
            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 {self.angle_count}, равносторонний {self.equilateral}, 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):
        if isinstance(perimeter,int) and isinstance(area,int) and isinstance(radius,int) and isinstance(diametr,int) and perimeter>0 and area>0 and radius>0 and diametr==2*radius and (color=='r' or color=='b' or color=='g'):
            super().__init__(perimeter, area, color)
            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.perimeter+self.area
        def __eq__(self,other):
            return self.radius==other.radius



    class PolygonList(list):
        def __init__(self,name):
            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):
            counts=1
            for item in self:
                print(f'{counts} многоугольник: {item.color}')
                counts+=1
        def print_count(self):
            print(len(self))



    class CircleList(list):
        def __init__(self, name):
            self.name=name
        def extend(self,iterable):
            super().extend(list(filter(lambda item: isinstance(item,Circle),iterable)))
        def print_colors(self):
            counts=1
            for item in self:
                print(f'{counts} окружность: {item.color}')
                counts+=1
        def total_area(self):
            print(sum(item.area for item in self))
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица Б.1 - Примеры тестовых случаев

№ 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__())
		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
			Верно

	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()		

2.	try: #неправильные данные для фигуры
	fig = Figure(-10,25,'g')
	except (TypeError, ValueError):
	print('OK')
	
	try:
	fig = Figure(10,-25,'g')
	except (TypeError, ValueError):
	print('OK')
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
			Верно

```
try:
    fig = Figure(10,25,-1)
except (TypeError, ValueError):
    print('OK')

try:
    fig = Figure(10,25,1)
except (TypeError, ValueError):
    print('OK')

try:
    fig = Figure(10,25,'a')
except (TypeError, ValueError):
    print('OK')

try:
    fig = Figure('a',25,'g')
except (TypeError, ValueError):
    print('OK')

try:
    fig = Figure(10,'a','g')
except (TypeError, ValueError):
    print('OK')

try:
    fig = Figure(0,25,'g')
except (TypeError, ValueError):
    print('OK')

try:
```

	fig = Figure(10,0,'g') except (TypeError, ValueError): print('OK')		

3.	try: #неправильные данные для многоугольника
	polygon = Polygon(-10,25,'g',4, True, 90)
	except (TypeError, ValueError):
	print('OK')
	
	try:
	polygon = Polygon(10,-25,'g',4, True, 90)
	except (TypeError, ValueError):
	print('OK')
	
	try:
	polygon = Polygon(10,25,1,4, True, 90)
	except (TypeError, ValueError):
	print('OK')
	
	try:
	polygon = Polygon(10, 25,'g',-4, True, 90)
	except (TypeError, ValueError):
	print('OK')
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
			Верно

	try: polygon = Polygon(10,25,'g',4, 2, 90) except (TypeError, ValueError): print('OK') try: polygon = Polygon(10,25,'g',4, True, -90) except (TypeError, ValueError): print('OK') try: polygon = Polygon('a',25,'g',4, True, 90) except (TypeError, ValueError): print('OK') try: polygon = Polygon(10,'a','g',4, True, 90) except (TypeError, ValueError): print('OK') try: polygon = Polygon(10,25,'a',4, True, 90)		

	except (TypeError, ValueError): print('OK') try: polygon = Polygon(10,25,'g','a', True, 90) except (TypeError, ValueError): print('OK') try: polygon = Polygon(10,25,'g',4, 'a', 90) except (TypeError, ValueError): print('OK') try: polygon = Polygon(10,25,'g',4, True, 'a') except (TypeError, ValueError): print('OK') try: polygon = Polygon(10,25,'g',1, True, 90) except (TypeError, ValueError): print('OK') try:		

```
    polygon = Polygon(10,25,'g',4, True, 0)
except (TypeError, ValueError):
    print('OK')

try:
    polygon = Polygon(0,25,'g',4, True, 90)
except (TypeError, ValueError):
    print('OK')

try:
    polygon = Polygon(10,0,'g',4, True, 90)
except (TypeError, ValueError):
    print('OK')

try:
    polygon = Polygon(10,25,'g',4, 0, 90)
except (TypeError, ValueError):
    print('OK')

try:
    polygon = Polygon(10,25,'g',4, True, 0)
except (TypeError, ValueError):
```

	print('OK')		

4.	try: #неправильные данные для окружности
	circle = Circle(-13, 13,'r', 2, 4)
	except (TypeError, ValueError):
	print('OK')
	
	try: #неправильные данные для окружности
	circle = Circle(13, -13,'r', 2, 4)
	except (TypeError, ValueError):
	print('OK')
	
	try: #неправильные данные для окружности
	circle = Circle(13, 13,-1, 2, 4)
	except (TypeError, ValueError):
	print('OK')
	
	try: #неправильные данные для окружности
	circle = Circle(13, 13,'r', -2, 4)
	except (TypeError, ValueError):
	print('OK')
	
	try: #неправильные данные для окружности
	circle = Circle(13, 13,'r', 2, -4)
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
			Верно

```
except (TypeError, ValueError):
    print('OK')

try: #неправильные данные для окружности
    circle = Circle('a', 13,'r', 2, 4)
except (TypeError, ValueError):
    print('OK')

try: #неправильные данные для окружности
    circle = Circle(13, 'a','r', 2, 4)
except (TypeError, ValueError):
    print('OK')

try: #неправильные данные для окружности
    circle = Circle(13, 13,'a', 2, 4)
except (TypeError, ValueError):
    print('OK')

try: #неправильные данные для окружности
    circle = Circle(13, 13,'r', 'a', 4)
except (TypeError, ValueError):
    print('OK')
```

```
try:    #неправильные данные для окружности
    circle = Circle(13, 13,'r', 2, 'a')
except (TypeError, ValueError):
    print('OK')

try:    #неправильные данные для окружности
    circle = Circle(0, 13,'r', 2, 4)
except (TypeError, ValueError):
    print('OK')

try:    #неправильные данные для окружности
    circle = Circle(13, 0,'r', 2, 4)
except (TypeError, ValueError):
    print('OK')

try:    #неправильные данные для окружности
    circle = Circle(13, 13,'r', 0, 4)
except (TypeError, ValueError):
    print('OK')

try:    #неправильные данные для окружности
    circle = Circle(13, 13,'r', 2, 0)
```

	except (TypeError, ValueError):
	print('OK')
	
	try: #неправильные данные для окружности
	circle = Circle(13, 13,'r', 2, 3)
	except (TypeError, ValueError):
	print('OK')		

5.	polygon = Polygon(10,25,'g',4, True, 90) polygon2 = Polygon(10,25,'g',4, True, 90) circle = Circle(13, 13,'r', 2, 4) circle2 = Circle(13, 13,'r', 2, 4) polygon_list = PolygonList(Polygon) try: #проверка списка многоугольников polygon_list.append(circle) except (TypeError): print('OK') try: polygon_list.append([polygon,polygon2]) except (TypeError):	OK OK OK OK OK OK2 OK2 OK2 OK2 OK OK	Верно

```
    print('OK')

try:

polygon_list.append(1)
except (TypeError):
    print('OK')

try:

polygon_list.append('pol
ygon')
except (TypeError):
    print('OK')

circle_list =
CircleList(Circle)
try: #проверка списка
окружностей

circle_list.extend(circle)
except (TypeError):
    print('OK')
finally:
    if(len(circle_list)==0):
        print('OK2')

try:

circle_list.extend([polyg
on, polygon2])
finally:
    if(len(circle_list)==0):
        print('OK2')

try:
```

	circle_list.extend([1,2])
	except (TypeError):
	print('OK')
	finally:
	if(len(circle_list)==0):
	print('OK2')
	
	try:
	
	circle_list.extend(['circle','circle2'])
	except (TypeError):
	print('OK')
	finally:
	if(len(circle_list)==0):
	print('OK2')
	
	try:
	
	circle_list.extend(polygon)
	except (TypeError):
	print('OK')
	finally:
	if(len(circle_list)==0):
	print('OK')		

6.	fig = Figure(10,25,'g')
	#проверка наследства
	try:
	if(isinstance(fig, Figure)):
	print('OK')
	except:
	pass
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
		OK
			Верно

```
polygon = Polygon(10,25,'g',4, True, 90)
try:
    if(isinstance(polygon, Figure)):
        print('OK')
except:
    pass
try:
    if(isinstance(polygon, Polygon)):
        print('OK')
except:
   pass

try:
    if(issubclass(Polygon, Figure)):
        print('OK')
except:
    pass

circle = Circle(13, 13,'r', 2, 4)
try:
    if(isinstance(circle, Figure)):
        print('OK')
except:
    pass
try:
    if(isinstance(circle, Circle)):
        print('OK')
except:
    pass
```

	try:
	  if(issubclass(Circle, Figure)):
	  print('OK')
	except:
	  pass
	
	polygon_list = PolygonList(Polygon)
	circle_list = CircleList(Circle)
	try:
	
	if(isinstance(polygon_list, list)):
	  print('OK')
	except:
	  pass
	try:
	
	if(isinstance(circle_list, list)):
	  print('OK')
	except:
	  pass		