**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

**Радиотехнический факультет (РТ)**

Отчёт по лабораторной работе № 3

По дисциплине

«Разработка интернет приложений»

|  |  |
| --- | --- |
| Проверил: | Выполнил: |
| Преподаватель кафедры ИУ-5 | студент группы РТ5-51Б |
| Гапанюк Ю.Е. | Лычагин Д.А. |
| Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_ | Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_ |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_ г. | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_ г. |

Москва, 2019

Задание

**Необходимо разработать программу, реализующую работу с классами.**

1. Программа должна быть разработана в виде консольного приложения на языке Python 3.
2. Все файлы проекта (кроме основного файла main.py) должны располагаться в пакете lab\_python\_oop.
3. Каждый из нижеперечисленных классов должен располагаться в отдельном файле пакета lab\_python\_oop.
   1. Абстрактный класс «Геометрическая фигура» содержит абстрактный метод для вычисления площади фигуры. Подробнее про абстрактные классы и методы Вы можете прочитать здесь - <https://docs.python.org/3/library/abc.html>
   2. Класс «Цвет фигуры» содержит свойство для описания цвета геометрической фигуры. Подробнее про описание свойств Вы можете прочитать здесь - <https://docs.python.org/3/library/functions.html#property>
   3. Класс «Прямоугольник» наследуется от класса «Геометрическая фигура». Класс должен содержать конструктор по параметрам «ширина», «высота» и «цвет». В конструкторе создается объект класса «Цвет фигуры» для хранения цвета. Класс должен переопределять метод, вычисляющий площадь фигуры.
   4. Класс «Круг» создается аналогично классу «Прямоугольник», задается параметр «радиус». Для вычисления площади используется константа math.pi из модуля math - <https://docs.python.org/3/library/math.html>
   5. Класс «Квадрат» наследуется от класса «Прямоугольник». Класс должен содержать конструктор по длине стороны.
   6. Для классов «Прямоугольник», «Квадрат», «Круг»:
      1. Определите метод "repr", который возвращает в виде строки основные параметры фигуры, ее цвет и площадь. Используйте метод format - <https://pyformat.info/>
      2. Название фигуры («Прямоугольник», «Квадрат», «Круг») должно задаваться в виде поля данных класса и возвращаться методом класса.

1. В корневом каталоге проекта создайте файл main.py для тестирования Ваших классов (используйте следующую конструкцию - <https://docs.python.org/3/library/__main__.html>). Создайте следующие объекты и выведите о них информацию в консоль:
   1. Прямоугольник синего цвета шириной 3 и высотой 2.
   2. Круг зеленого цвета радиусом 5.
   3. Квадрат красного цвета со стороной 5.

Исходный код

Main

**from** lab\_python\_oop.Rectangle **import** Rectangle  
**from** lab\_python\_oop.Square **import** Square  
**from** lab\_python\_oop.Circle **import** Circle  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 r1 = Rectangle(3, 2, **"Синий"**)  
 print(r1)  
 s1 = Square(5, **"Красный"**)  
 print(s1)  
 c1 = Circle(5, **"Зелёный"**)  
 print(c1)

Square

**from** lab\_python\_oop.Figure **import** Figure  
**from** lab\_python\_oop.ColorF **import** ColorF  
**from** lab\_python\_oop.Rectangle **import** Rectangle  
  
**class** Square(Rectangle):  
 **def** \_\_init\_\_(self, d, color, name=**"квадрат"**):  
 self.d=d  
 self.color=color  
 self.name=name  
 **def** \_\_repr\_\_(self):  
 **return '{} {} со стороной {} '**.format(self.color, self.name, self.d)  
 **def** S(self, d):  
 **return** self.d\*self.d

Rectangle

**from** lab\_python\_oop.Figure **import** Figure  
**from** lab\_python\_oop.ColorF **import** ColorF  
  
**class** Rectangle (Figure):  
 **def** \_\_repr\_\_(self):  
 **return '{} {} высотой {} и шириной {}'**.format(self.color, self.name, self.height, self.width)  
 **def** \_\_init\_\_(self, width, height, color, name=**"прямоугольник"**):  
 self.width = width  
 self.height = height  
 self.color=color  
 self.name=name  
 **def** S(self, width, height):  
 **return** self.width \* self.height

Figure

**from** abc **import** ABCMeta, abstractmethod  
**class** Figure(object):  
 \_\_metaclass\_\_ = ABCMeta  
@abstractmethod  
**def** S(self, x, y):  
 **pass**

Color

**class** ColorF:  
 **def** \_\_init\_\_(self, color):  
 self.color = color  
 @property  
 **def** svoistv(self):  
 **return** self.color

Circle

**from** lab\_python\_oop.Figure **import** Figure  
**from** lab\_python\_oop.ColorF **import** ColorF  
**import** math  
**class** Circle(Figure):  
 **def** \_\_init\_\_(self, R, color, name=**"круг"**):  
 self.R = R  
 self.color=color  
 color=ColorF(self.color)  
 self.name=name  
  
 **def** S(self,R):  
 **return** self.R \* self.R\*math.pi  
 **def** \_\_repr\_\_(self):  
 **return '{} {} радиусом {} '**.format(self.color, self.name, self.R)

Скриншоты

