|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет** **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Отчёт по лабораторной работе №2 по курсу**

**«Разработка Интернет-Приложений»**

**Тема работы: "Классы, методы, поля. Наследование, полиморфизм, инкапсуляция. Модули. Работа с пакетным менеджером pip**.**"**

Выполнил: Забурунов Леонид, РТ5-51Б

Проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

12 сентября 2020 г.

ЗАЧТЕНО / НЕ ЗАЧТЕНО\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

# Цель лабораторной работы.

Изучение основных конструкций и возможностей ЯП Python при объектно-ориентированном подходе.

# Задание на лабораторную работу.

Для лабораторной работы №2 необходимо разработать консольное приложение, включающее в себя несколько модулей, представляющих из себя классы с общим абстрактным прародительским классом, тема классов – геометрические фигуры. Требуется разработать абстрактный класс, а так же классы для трёх основных геометрических форм – круга, прямоугольника и квадрата – с реализацией функционала:

1. Хранения информации об измерениях фигуры и её цвете для экземпляра класса;
2. вычисления площади фигуры для экземпляра класса;
3. представления экземпляра класса в удобном для чтения виде;
4. статического поля, сообщающего о типе фигуры, для каждого класса.

# Ход выполнения лабораторной работы.

Начнём с самого абстрактного класса. Реализуем весь его функционал в модуле *shape.py*:

**from** abc **import** ABC, abstractmethod  
  
**class** Shape(ABC):  
 *# Абстрактный метод вычисления площади* @abstractmethod  
 **def** Square(self):  
 **pass** *# Виртуальный метод возвращения типа* **def** GetType(cls):  
 **return** cls.shapeType.lower()  
  
 *# Виртуальный метод форматированного представления всех фигур* **def** \_\_repr\_\_(shapeObject):  
 **return "Данная фигура является {0}ом, имеет {1} цвет, а её площадь равняется {2} на основании введённых числовых параметров:"**.format(shapeObject.GetType(), shapeObject.color.Color, shapeObject.Square())  
  
 *# Статический метод проверки корректности ввода числовых значений (для измерений)* @staticmethod  
 **def** CheckNumValue(num):  
 **try**:  
 *# Проверяем возможность перевода в численный формат* value = float(num)  
 *# Если возможно, то возвращаем полученное значение без изменений* **return** value  
 **except** ValueError:  
 *# Если преобразование типа невозможно, то требуем повторного ввода до тех пор, пока не получим число* isNum = **False  
 while not** isNum:  
 print(**"Ошибка получения величины. Введите значение ещё раз"**)  
 **try**:  
 value = float(input())  
 isNum = **True  
 except** ValueError:  
 **pass  
 return** value

Следующим шагом создаём класс для работы с цветом фигур, *color.py*:

**class** Color:  
 *# Статический метод проверки корректности ввода строки цвета* @staticmethod  
 **def** CheckColorValue(value):  
 isNum = **not** value.isalpha()  
 *# Если введенно некорректное значение, то требуем повторного ввода до тех пор, пока не получим буквенную строку* **while** isNum:  
 print(**"Ошибка получения цвета. Введите значение ещё раз"**)  
 value = input()  
 isNum = **not** value.isalpha()  
 **return** value  
  
 **def** \_\_init\_\_(self, colorValue):  
 self.color = Color.CheckColorValue(colorValue)  
  
 *# Св-во цвета* @property  
 **def** Color(self):  
 **return** self.color.lower()  
  
 @Color.setter  
 **def** Color(self, colorValue):  
 colorValue = self.CheckColorValue(colorValue)  
 self.color = colorValue

Теперь создаём классы для самих фигур:

1. Прямоугольник, *rectangle.py*:

**from** lab\_python\_oop.shape **import** Shape  
**from** lab\_python\_oop.color **import** Color  
  
**class** Rectangle(Shape):  
 shapeType = **"Прямоугольник"** @classmethod  
 **def** GetType(cls):  
 **return** Shape.GetType(cls)  
  
 **def** \_\_init\_\_(self, argsList):  
 print(**"Записываем ширину..."**)  
 self.width = Shape.CheckNumValue(argsList[0] **if** len(argsList) >= 1 **else '.'**)  
 print(**"Записываем высоту..."**)  
 self.height = Shape.CheckNumValue(argsList[1] **if** len(argsList) >= 2 **else '.'**)  
 print(**"Записываем цвет..."**)  
 self.color = Color(argsList[2] **if** len(argsList) >= 3 **else '0'**)  
 print(**"Запись для прямоугольника завершена!"**)  
  
  
 **def** Square(self):  
 **return** self.width \* self.height  
  
 **def** \_\_repr\_\_(self):  
 **return** Shape.\_\_repr\_\_(self) + **"\nШирина: {0} \nВысота: {1}"**.format(self.width, self.height)

1. Круг, *circle.py*:

**from** lab\_python\_oop.shape **import** Shape  
**from** lab\_python\_oop.color **import** Color  
**from** math **import** pi **as** PI  
  
  
**class** Circle(Shape):  
 shapeType = **"Круг"** @classmethod  
 **def** GetType(cls):  
 **return** Shape.GetType(cls)  
  
 **def** \_\_init\_\_(self, argsList):  
 print(**"Записываем радиус..."**)  
 self.radius = Shape.CheckNumValue(argsList[0] **if** len(argsList) >= 1 **else '.'**)  
 print(**"Записываем цвет..."**)  
 self.color = Color(argsList[1] **if** len(argsList) >= 2 **else '0'**)  
 print(**"Запись для круга завершена!"**)  
  
 **def** Square(self):  
 **return** PI \* self.radius \* self.radius  
  
 **def** \_\_repr\_\_(self):  
 **return** Shape.\_\_repr\_\_(self) + **"\nРадиус: {}"**.format(self.radius)

1. Квадрат, *quadrate.py*:

**from** lab\_python\_oop.shape **import** Shape  
**from** lab\_python\_oop.rectangle **import** Rectangle  
**from** lab\_python\_oop.color **import** Color  
  
  
**class** Quadrate(Rectangle):  
 shapeType = **"Квадрат"** @classmethod  
 **def** GetType(cls):  
 **return** Shape.GetType(cls)  
  
 **def** \_\_init\_\_(self, argsList):  
 print(**"Записываем cторону..."**)  
 self.dim = Shape.CheckNumValue(argsList[0] **if** len(argsList) >= 1 **else '.'**)  
 print(**"Записываем цвет..."**)  
 self.color = Color(argsList[1] **if** len(argsList) >= 2 **else '0'**)  
 print(**"Запись для квадрата завершена!"**)  
  
  
 **def** Square(self):  
 **return** self.dim \* self.dim  
  
 **def** \_\_repr\_\_(self):  
 **return** Shape.\_\_repr\_\_(self) + **"\nСторона: {}"**.format(self.dim)

Все рабочие модули созданы, последний шаг – основная программа для проверки работы, в которую так же будет решён вопрос последнего требования к лабработе – загрузки библиотеки через pip. *Main.py*:

**from** lab\_python\_oop.rectangle **import** Rectangle  
**from** lab\_python\_oop.circle **import** Circle  
**from** lab\_python\_oop.quadrate **import** Quadrate  
**import** numpy **as** np

**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 print(**"Забурунов Леонид Вячеславович, группа РТ5-51Б"**)  
 print(**"Лабработа #2"**)  
 print(**"Программа проверяет функционал созданных по заданию лабораторной работы модулей."**)  
 print(**"Вам будет предложено ввести данные для прямоугольника, круга и квадрата."**)  
 print(**"Ввод для каждого объекта осуществляется в одну строку, через пробел."**)  
 print(**"Введите данные прямоугольника: "**)  
 rctg = Rectangle(input().split())  
 print(**"Введите данные круга: "**)  
 crc = Circle(input().split())  
 print(**"Введите данные квадрата: "**)  
 qdr = Quadrate(input().split())  
 print(**''**, **'Итоги ввода: '**, rctg, crc, qdr, sep = **'\n\n'**)  
 print(**'\n'**, np.ones((3, 20), float))

# Результаты работы

